Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1983

Nonparametric statistics test software package.

O'Brien, Philip J.,Jr.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/19921

Downloaded from NPS Archive: Calhoun

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

T218327

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NONPARAMETRIC STATISTICS TEST SOFTWARE PACKAGE

by

Philip J. O'Brien, Jr.

September 1983

Thesis Advisor: Stephen Paek

Approved for Public Release; Distribution Unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Nonparametric Statistics Test Software Package

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis
September 1983

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Philip J. O'Brien, Jr.

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943

12. REPORT DATE
September 1983

13. NUMBER OF PAGES
249

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for Public Release, Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Nonparametric Tests
Binomial
Quantile
Distribution Free Tests
Wilcoxon
Kolmogorov
Smirnov

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis provides the computer software to perform 24 of the most common nonparametric tests, and a text explaining how and why to use that software.

Nonparametric tests are valuable to experimenters and operations analysts for three reasons: ease of explanation to non-statisticians, simplicity of computation, and applicability to data

sets which cannot be analyzed by parametric tests. Reasons for unsuitability of parametric test include data of nominal or ordinal level of measurement, lack of common variance, or lack of normal distribution of the underlying population.

There are two programs provided: Lochinvar, an interactive Pascal program, and Crunch, a Fortran program. Lochinvar is designed to prompt and screen a user's entry of data and options. Crunch performs calculations; it is suitable for use in whole or part as a subcomponent of other programs.

Approved for Public Release, Distribution Unlimited

Nonparametric Statistics Test Software Package

by

Philip J. O'Brien, Jr.
Major, United States Marine Corps
A.B., Stanford University, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September, 1983

ABSTRACT

This thesis provides the computer software to perform 24 of the most common nonparametric tests, and a text explaining how and why to use that software.

Nonparametric tests are valuable to experimenters and operations analysts for three reasons: ease of explanation to non-statisticians, simplicity of computation, and applicability to data sets which cannot be analyzed by parametric tests. Reasons for unsuitability of parametric test include data of nominal or ordinal level of measurement, lack of common variance, or lack of normal distribution of the underlying population.

There are two programs provided: Lochinvar, an interactive Pascal program, and Crunch, a Fortran program. Lochinvar is designed to prompt and screen a user's entry of data and options. Crunch performs calculations; it is suitable for use in whole or part as a subcomponent of other programs.

TABLE OF CONTENTS

I. INTRODUCTION 8
II. COMPARISON OF PARAMETRIC VERSUS NONPARAMETRIC TESTS 10
III. THE COMPUTER PACKAGE 18
IV. ROSTER OF AVAILABLE NONPARAMETERIC TESTS ...26
V. BINOMIAL TEST.................................31
VI. QUANTILE TEST................................36
VII. MANN WHITNEY TEST...........................41
VIII. COX STUART TEST FOR TREND 44
IX. SIGN TEST 48
X. MCNEMAR TEST FOR SIGNIFICANCE OF CHANGE 52
XI. CHI-SQUARE TEST FOR INDEPENDENCE (R*C CONTINGENCY) 57
XII. CHI-SQUARE GOODNESS OF FIT TEST 62
XIII. WILCOXON SIGNED RANK TEST 66
XIV. NONPARAMETRIC CORRELATION 70
XV. NONPARAMETRIC REGRESSION 75
XVI. MONTONIC REGRESSION 80
XXVI. MEDIAN TEST 84
XVIII. KRUSKAL WALLIS TEST 89
XIX. SQUARED RANK TEST FOR EQUAL VARIANCE 93
XX. HARTLEY TEST FOR EQUAL VARIANCE 97
XXI. FRIEDMAN TEST AND DURBIN TEST 100
XXII. QUADE TEST 105
XXIII. COCHRAN TEST FOR RELATED OBSERVATIONS 109
XXIV. KOLMOGOROV TEST 114
XXV. LILLIEFORS TEST 119
XXVI. SHAPIRO WILK TEST FOR NORMALITY 122
XXVII. SMIRNOV TEST...............................125
XXXVIII. CRAMER VON MISES TEST 128
LIST OF REFERENCES 131

APPENDIX A TUTORIAL: LINKING TO AND USING THE COMPUTER PACKAGE 132
APPENDIX B THE PROGRAM LOCHINVAR 139
APPENDIX C THE DATA FILE 144
APPENDIX D THE OPTIONS FILE 149
APPENDIX E THE PROGRAM CRUNCH 153
APPENDIX F FORTRAN SOURCE CODE: CRUNCH 158
APPENDIX G PASCAL SOURCE CODE: LOCHINVAR 211
INITIAL DISTRIBUTION LIST 248

ACKNOWLEDGEMENT

The greatest debt which I, and contemporary students of nonparametric statistics, owe is to W. J. CONOVER, who wrote a lucid and succinct text on the subject. This both encouraged and facilitated study of nonparametric tests; my thesis would have been immeasurably more difficult without his book.

My thesis advisor, LTC Stephen PAEK, USA, taught me the subject matter and channeled my efforts towards making a usable computer package. Both he and my second reader, CDR Charles HUTCHINS, USN, made numerous suggestions for improved syntax, sentence structure and spelling throughout this text. With cunning and perseverance, I have managed to sneak a few errors past their watchful eyes. I not only claim them as my own fault, but leave their correction to the student as an exercize.

My father began drilling me in mathematics before I went off to kindergarden. Both he and my mother made sacrifices to see that I was expensively educated. More importantly, they encouraged me to use at least some portion of my God-given talents.

I. INTRODUCTION

This thesis is intended to help non-statiticians use nonparametric tests to their fullest. The text explains when nonparametric tests are preferable to parametric tests, how to choose the most appropriate test, and how to use the nonparametric software package.

Parametric tests are often the preferred form of statistics because of their acceptance in the academic world, the availability of computer support, and flexibility in model building. Nonparametric tests provide a valuable alternative to parametrics, because they may be applied to data which, for any one of a number of reasons, cannot be analyzed with parametric tests. The simplicity of nonparametric tests often make the test statistics easier to compute and test results easier to explain. Section II gives a detailed comparison of parametric versus nonparametric tests.

When confronted with the same hypothesis, various parametric tests give consistent results because each test extracts the same information from the data set. Nonparametric tests may yield varying results with the same data, because different tests extract different information from the data set. To overcome this potential problem, the most appropriate nonparametric test should be chosen to analyze the data set. Section IV gives guidance on selecting the best test.

In order to make using nonparametric tests as easy as possible, two programs were written to perform the most common 24 nonparametric tests. A general description of how to use the programs is contained in section III. Sections V through XXVIII contain descriptions of each of the tests,

and detailed instructions on their use. The appendices contain more detailed information about the programs. Appendix A is a tutorial which depicts use of the programs to perform a test on a data set. Appendices B through E describe the two programs and two files. Appendix G contains the interactive Pascal program, Lochinvar, while Appendix F contains the Fortran program Crunch.

II. COMPARISON OF PARAMETRIC VERSUS NONPARAMETRIC TESTS

This section contrasts the nonparametric statistical test with parametric statistics. To do this, there is a short summary addressing various levels of measurement, and a brief discussion on the relationship between the normal distribution and interval data. Finally there is a comparison of the advantages of the two classes of tests. This will help the reader decide whether to use the tests of this text.

A. LEVELS OF MEASUREMENT

Statisics manipulate raw data to extract whatever information that data contains. Level of measurement indicate how much information is contained in the data, and how that information may be manipulated. Although formally there are four levels of measurement, as defined by Stevens [Ref. 2], there are only three which matter when choosing the test: nominal, ordinal, and interval. As a consistent rule any information contained in data of a lower level of measurement is contained in data of a higher level; any manipulation permitted on a lower level is permitted on a higher level.

Nominal data elements contain very little information: only membership in a subset of the total data set can be determined. Manipulation of the nominal data sets is limited to counting the number of elements in each subset.

Ordinal data elements contain information relative to other elements: element A is either less than, greater than, or equal to each other element in the set. However, the question "greater than by how much" cannot be answered. The data set may be ranked.

Interval data elements contain more information. Not only can element A be compared to other elements, but inequalities can be quantified; e.g., A>B implies that there exists a k such that A=B+k, and that the k can be found by k=A-B. The data set can be manipulated with normal arithmetic operations, such as addition, division, or raising to a power.

B. LEVEL OF MEASUREMENT AND DISTRIBUTION ASSUMPTIONS

The normal distribution is completely specified by two parameters: expected value and variance. In estimating those parameters form a data set, arithmetic operations are used, and intervals are treated as though they were meaningful. For example, the expected value is estimated by the sample mean, which is found by summing the data elements and dividing by the number of elements. The variance is found by summing the squares of the intervals between each data element and the sample mean, and dividing by the number of elements minus one. Because of the use of arithmetic operations and intervals in computing its parameters, the normal distributon seems to require interval data sets.

Parametric tests assume that a data set contains elements with an interval level of measurement drawn from a normally distributed population. Comparison of two data sets usually require the assumption of equal variance in the two populations from which the samples were drawn. There is a fair amount of literature concerning how valid parametric tests are when any of these assumptions are violated; a large, vocal group of statisticians would say that such tests produce valid results even when the assumptions are decidedly not met. But meeting all of the assumptions of parametric tests are the guarantee of validity when using such test.

Nonparametric tests make no assumptions about the underlying distribution of the population from which the data is drawn. Additionally, there are nonparametric tests to analyze data with nominal level of measurement or higher. Because some nonparametric tests require only nominal or ordinal data, and don't make assumptions about population distributions, the power of those tests may be less than that of parametric tests-- assuming that the data set justified use of parametrics in the first place.

C. ADVANTAGES AND DISADVANTAGES OF CLASSES OF TESTS

The advantages and disadvantages of parametric tests are the mirror image of those of nonparametric test. Therefore, the advantages of each class of tests is given.

1. Advantages of Parametric Tests

The general advantages of parametric tests are wide acceptance of methodology, consistency of results, and capability to handle multiple variables in systematic model building.

a. Wide Acceptance

The wide acceptance of parametric test has a number of advantages. Statistical software packages contain a very complete set of parametric tests, and only a small---although increasing over time--number of nonparametric tests. Table of the theoretical distributions needed to judge test statistics (the normal, student's t, chi-square and F) are extensive and ubiquitous. The terms of the test are known to a majority of persons trained in the social or physical sciences; whether the terms are fully understood may not influence the value of their recognition.

b. Consistency of Results

Different parametric tests give the same conclusions, given the same null and alternative hypotheses. Parametric tests all extract the same information from the data, and merely rearrange it for different purposes. The parametric distributions are all derived from the normal.[1] With related test statistics and similar test distributions for critical values, it is not surprising that different techniques give consistent answers to the same question.

Nonparametrics may provide different results to the same question on the same data set, because different tests extract different types of information from the data set. For example, with paired data elements, the sign test considers only which member of the pair is larger; the signed rank test needs to know not only which is larger, but also the magnitude of that difference. Consistency of results is therefor a possible advantage for parametric tests. However, this advantage can be minimized by selecting the best (most powerful) nonparametric test possible, as discussed in a later section.

c. Versatility in Model Building

The final, and probably most important, advantage of parametrics is their versatility in explaning very complex interactions in data. Analysis of variance can test differences in treatment effects, multiple blocking effects, and interaction among treatments and blockings. Regression can be made to do analysis of variance, (by using nothing but dichotomous variables), but is even more flexible, by building models using variables with continuous values rather than only a fixed number of treatment levels.

---

[1] The t is a normal distribution in which the lack of knowledge of the population expected value is corrected for by lowered kurtosis; the chi-square distribution of degrees of freedom n is the sum of squares of n standard normal distributions; the F is the ratio of two chi-squares.

This versatility of parametrics is based on the very strong assumptions made about the populations from which the data is drawn: literature beyond the elementary statistics textbook level contains many ways to determine if the assumptions are met, and what to do when they are not met.

Nonparametric tests, on the other hand, assume nothing about the underlying distribution, and often work with nominal or ordinal data. Not surprisingly, only very straight-forward answers can be given for any one test. A clever application of a sequence of tests to the same data set may, however, yield answers to a number of different questions, and provide some inferences on interaction.

2. Advantages of Nonparametric Tests

The advantages of nonparametric tests are simplicity and applicability.

a. Simplicity

Nonparametric tests' simplicity allows for relatively easy calculation of the test statistic and critical value for a null hypothesis. This simplicity also allows nonparametric tests to be explained to, and defended in front of, persons who are not trained in statistics.

The test statistic for most nonparametic tests can be calculated by hand. For example, a set of 20 pairs of data elements (X,Y) can be compared for equality of expected value (E(X)=E(Y)) by either the t-test or the nonparametric sign test. The t-test needs sample means and variances, so some arithmetic operations are required: four sums of 20 terms each, 40 squaring of numbers, four divisions. In contrast, the sign test requires counting in how many pairs the X element is greater than the Y element. The contrast is not always so one sided; as the amount of information extracted from the data increases, so does complexity of computation. For example, rank-order nonparametric tests

require ranking of data elements, which can become tedious in large data sets. However, the process of ranking tends to be self-correcting, because an unranked or misranked data value will stand out in a simple sequential check of the ranks. Arithmetic mistakes in squaring, summing, et cetera, may not be so obvious unless the operation is redone. When computations are made by hand, therefore, nonparametric test statisistics are easier to find.

Since hand-held calculators are even more ubiquitous than parametric tables, this advantage of nonparametric test is less important than it once was. If the number of keystroke errors is proportional to the number of numbers entered in the calculator, though, the simplicity of nonparametric tests may still be an advantage.

Simplicity could also be an advantage in the rare case when tables of distribution are not available. Many nonparametric test distributions, especially the simpler tests, rest on the binomial distribution. For small sample sizes, the binomial can be calculated by hand; with a small calculator, a distribution for sample size of 20 or 30 can easily be calculated. The normal, student's t, chi-square and F distributions lack closed forms for their cumulative distribution functions. Numerical integration--even if the density functions were remembered or available--would be challenging with a calculator (since all but the normal require one or more gamma functions) and probably is beyond the realm of paper and pencil mathematics.

Simplicity is also a virtue in explaining test results. Comparison and rankings are fairly easy to follow (although sometimes difficult to perform), while the arithmetic operations of parametric tests are less transparent. Most non-statisticians have had experience in tossing coins, rolling dice or some games of chance. Analogies can be made from those experiences to the evaluation of test results.

In the previous example of 20 pairs (X,Y), with the sign test producing a test statistic of 4 (i.e., in four pairs, X>Y), an analogy could be made to the odds that a coin were fair if it came up heads four times in 20. Explaining that same data set with the t-test would require mention of standard deviations and pooled variances, and perhaps drawing of overlapping bell-shaped curves. An audience predisposed to disagree with the test results might be hard to convince with parametric tests.

b. Applicability

Nonparametric tests can be used on a larger number of data set than can parametic tests, because of the lack of assumptions about the distribution of the underlying population, and the ability to handle ordinal or interval data. This advantage is more vital in the "real world" than in academia, for three general reasons:

(1) An analyst is sometimes handed data sets to do something with; for any one of a number of reasons, parametric tests may not applicable. Most commonly, the form in which the data was collected in may fail to meet the underlying requirements of parametrics, but meets the requirements of nonparametrics.

(2) The cost of gathering interval data is prohibitive, while ordinal or nominal data can be gathered at low cost. For example, on a field exercise, uncalibrated eyes can generally judge distances greater than or less than, but some sort of instrumentation must be used to determine the exact magnitude of the difference. The expensively equipped national test ranges are a response to the need for very accurate interval level of measurement data. Laboratories, too, are often equipped for fine metrics. But in field or operational tests, ordinal level of measurement may be the highest level available.

(3) The experiment was designed to gather interval data, from populations with common variance, but something went wrong. Instrumentation sometimes fails. Even if the measurement is accurate, variances may not be common or the data may not be transformable to look normal. Rather than giving up, or repeating the experiment, nonparametric tests may answer some questions originally to be answered by analysis of variance or regression.

III. THE COMPUTER PACKAGE

A. GENERAL

This section gives a brief description of the computer package. Tests are described in sections V through XXVIII, while the Appendices describe the programs and files in detail.

The computer package consists of two computer programs, Lochinvar and Crunch. Also involved are three types of CMS files: data files and an options file, written by Lochinvar to be read by Crunch; and output files, written by Crunch, containing test results.

B. LOCHINVAR

The program Lochinvar (LOad CHoices and INput VARiables) is the user's main contact with the programs. It is interactive, prompting and screening the user's entries. Its purpose is to write two types of files needed by the program Crunch: the data file, and the option file.

1. Data Entry

Lochinvar allows the user to input data in two separate ways. The user must decide which one to use before starting the program.

a. Method I: Data Values into a Data File

When the data is numerical, the user should strongly consider this method. Lochinvar prompts entry of data set and variable names, and of the data set values themselves. All that information is sent to a file on the user's A disk; the user chooses one of six files, "lochi a",...,"lochi f", into which the data is written. Unless

erased or over-written, the data will be available for repeated use.

All 24 of the data tests in this text can compute test statistics from data files. However, a data file itself is not enough to perform a test; the option file is needed to reference the data file and communicate the choice of test and test parameters to Crunch. After a data file is written, Lochinvar prompts the writing of the options file, as described in subsection 2, below.

b. Method II: Cell Counts in the Options File

When the data is non-numerical, this is the user's only option; it may be more convenient even if the data is numerical. Four of the nonparametric tests handle nominal data--binomial, McNemar, chi-square test for independence, and Cochran. Instead of entering all the data values, the user may run one of these tests by entering the appropriate count of elements which fall into a test's subcategories. For example, the binomial test can be run on a large data set by simply entering two numbers: the count of "successes" and the count of "failures".

In this method of data entry, Lochinvar prompts entry of cell counts at the same time it is prompting the test selection and choice of parameter. All that information is written to the options file; no data file is written.

2. The Options File

The purpose of the options file is to tell Crunch everything it needs to know to perform a nonparametric test. In all cases, the options file contains the user's selection of which test to be performed, the parameters to be used in that test, and to which CMS output file the output should be written. If the data values are to be used (Method I), the option file states which data file Crunch should read. If cell counts are used (Method II), the options file contains those counts, as well as the data set name.

The options file is described in detail, with an example, in Appendix D.

3. Using Lochinvar

In CMS, type "LOCHI". The program will display a sequence of questions; the user's response will depend on the method of data enty and test selection.

a. Use of Previously Enter Data

The program asks "do you wish to perform a test on a previously entered data set?", i.e., one of the six CMS data files "lochi a",...,"lochi f".

If the answer is yes, then Lochinvar asks which data set to use. The program reads the data set's name, and displays it to the user, and asks "Is this the right data set?" If yes, the user indicates so, and the test selection is next. If no, the the user must choose another file.

If the answer to the question of previously enter data is no, then Lochinvar asks whether data entry is by data values (Method I) or cell counts (Method II). If the user wants to enter cell counts, the next step is to select a test. If the user wants to enter a data set, the program asks:

(1) To which file the input data is to be written

(2) What is the data set name, number of variables and variable names.

The program will then prompt entry of data values. If there are two variables in the data set, the user will be offered the choice of entering data pair by pair; this will only work if there are an equal number of elements in each of the two variables. Normally, all the data elements of the first variable will be entered together, then all of the second variable, et cetera. After entry of each variable, Lochinvar will offer the user the chance to check, change or add data elements to the values just entered.

After the data set's information has been entered, Lochinvar writes it to the specified file. Writing of the options file is next.

b. Selection of the Test to be Performed

After the choice of data entry has been given, Lochinvar offers a menu of tests from which the user may choose. If cell counts was the chosen method of data entry, four tests are offered. If a data file is to be used, the choice set of test offered is determined by:

(1) the number of variables in the data set

(2) whether the varibles have an equal number of elements

Lochinvar will not offer a test unless the data set contains at least the minimum number of variables needed to perform the test.

Some nonparametric tests require that data elements of one variable be related to corresponding elements of other variables, i.e., that the data set be paired or blocked. Lochinvar will not offer such a test unless there are the same number of elements in each of the data set's variables. However, variables can be equal sized but not paired or blocked. The user should not choose an inappropriate test just because Lochinvar offers it.

c. Output File Selection

The user chooses among six possible output files, "u listing",..."z listing". The files will be written to user's A disk after execution of the program Crunch. Previous contents of the designated file will be overwritten after Crunch runs.

d. Test Parameter Entries

Each test has different entries required. See the section on the desired test, sections V through XXVIII for specifics.

C. CRUNCH

1. Executing the Nonparametric Test

After the options file has been written, the user executes the test by typing "CRUNCH" while in CMS.

2. Actions of the Program Crunch

Crunch reads the options file, and when necessary, the designated data file. It performs the nonparametric test requested, and reminds the user with a display on the screen to which output file the results have been written. Crunch actually writes three files:

a. The Output File

The output files are named "u listing", ..., "z listing". At the top of of the file is the label, consisting of the name of the data set, the number of variables in the data set, the number of variables used in the test (if less than the number in the whole data set), and the names of the variables used in the test. Following the label is the name of the test performed. The rest of the output file's format varies, depending on the test chosen. In general, the output file contains only the most salient results of the computations. All output files contain either a test statistic, or a test statistic and the empirical level of significance for the desired null hypothesis.

b. The Problem Listing File

If there are any problems encountered by the computer that result in Fortran error or warning messages, such messages will be written to the CMS file "problem listing".

c. The Nptest Listing File

This file contains intermediate computations of the program as it works towards the test statistic. For example, the Smirnov test requires finding differences in probabilities of two empirical distributions. Since the

test statistic is simply the largest absolute difference, the output file will only list the largest positive and most negative difference, as well as the test statistic. "Nptest listing", among other things, will include each difference between the empirical cumulative distribution functions (cdf's). This listing file is not arranged to be attractive or suitable for submission, but it is labelled, and may be of some use in understanding the operations of Crunch.

D. ADDITIONAL FEATURES OF THE PACKAGE

To avoid overwriting input or output files, six of each were provided. If more are needed, the user can rename some of the existing files.

Lochinvar is a convenient way to enter data by hand. When data is generated from other programs or read from disk or tape, the user may want to create his own data file rather than use Lochinvar. Detailed instructions for this are contained in Appendix C.

Crunch is written in Fortran. Users who are experienced programmers may want to modify it to suit their own tastes, or adapt portions of it into their own programs. An explanation of the program is contained in Appendix E. The program is listed in Appendix G.

E. CONSIDERATIONS IN DATA ENTRY

Lochinvar is written in Pascal, a convenient language for interactive programs, but a language which may be unfamiliar to many users. This section provides sufficient insight to Lochinvar so that the user can efficiently communicate choices to the program.

1. Entry of Numbers

When entering numerical data, Lochinvar requires that the first character read is either a numeral or a minus sign followed by a numeral. Therefore, to enter a number between minus one and plus one, precede the decimal with a "0".

When entering more than one number at a time, leave a space (not a comma) between the numbers.

If Lochinvar expects a number and gets anything else, it will halt with an error. This is a convenient way to get out of the program quickly, but anything entered on that run of Lochinvar will be lost. The one exception is when entering a data set of data values and an options file in a single run. After the data set is entered, the program writes it to the designated file, and asks "now, do a test with this data?" Any subsequent errors will not effect the data file itself, just erase all additional entries.

2. Entry of Characters

When the program prompts for a single letter response, e.g., "enter Y or N", enter "Y" or "N" and press the enter key. The program expects and will read only the first letter. Single letter entries are contained in "repeat...until" loops which will not allow the user to procede until one of the offered choices is made.

Do not use the alternate character (apl) setting on the keyboard while using Lochinvar.

When Lochinvar prompts for the data set name or variable names, the user must enter from one to 48 characters. These names are one of the few things for which the program does not offer a chance to correct an error. The user must be careful, checking the names prior to hitting the "enter" key. If an error is made anyway, the data or options file can be corrected using Xedit in CMS.

3. Minimum Entry

If the program prompts for an entry, enter at least one numeral or character. The interactive read commands are in "do while not eoln " (end of line) loops; if the enter key is pressed with nothing to enter, Lochinvar will halt with an error, because it attempted to read beyond the end of the line. As with numerical entry errors, previous entries will be lost.

IV. ROSTER OF AVAILABLE NONPARAMETRIC TESTS

A. CRITERIA FOR SELECTING THE MOST APPROPRIATE TEST

In selecting a nonparametric test, the user must consider three criteria: purpose of the test, data set level of measurement, and whether the data elements are related (i.e., paired or blocked).

B. CRITERION I: PURPOSE OF TEST

Nonparametric tests can be successfully used to test for questions of location, independence/interaction, and distribution. Table 1 shows a matrix of the nonparametric test with level of measurement (LOM) on the horizontal axis and general purpose on the vertical.

Location usually refers to whether the expected value of one population is equal to that of another, based on sample variables. Other purposes are testing for proportion of successes or for value of a single population's quantile. Table 2 lists the location tests with their null hypotheses, to help the user narrow the choice set.

Independence of two samples refer to statistical independence, i.e., correlation of zero. Interaction may be a before/after effect, or prediction of one variable by another. Table 3 lists these tests by their specific null hypotheses.

Some tests for distribution check for equal variance among variables. Other tests check a single variable's empirical cumulative distribution function (ecdf) against a theoretical distribution, with either hypothesized or estimated parameters. Still other tests compare the empirical

TABLE 1

Nonparametric Tests

Purpose, Minimum # Vars:	Levels of Measurements:		
	Interval	Ordinal	Nominal
LOCATION:			
1 var		Quantile	Binomial
2 var	Wilcoxon*	Sign* Mann-Whitney	
≥2 var	Quade*	Median Friedman* Kruskal-Wallis	Cochran
INDEPENDENCE/ INTERACTION:			
1 var		Cox-Stuart	McNemar
2 var	Correlation NP Regression Monotone Regression	Cox-Stuart*	R*C Contingency (chi-square)*
DISTRIBUTION:			
1 var	Kolmogorov (N<31) Goodness of Fit (chi-square,N>14) Wilkes-Shapiro (N<31) Lilliefors (N<31)		
2 var		Smirnov Cramer-von Mises	
≥2 var	Squared Rank Hartley		

* indicates that variables must be paired or blocked

cdf's of two variables against each other. Table 4 has a list of these tests, with their hypotheses, levels of measurement, and other considerations.

The decision matrix indentifies tests suited to each of these purposes. The specific section on the test should be checked to be sure that the test will do what is desired.

TABLE 2

Nonparametric Tests for Location

HYPOTHESIS	NAME	# VARS	LOM	SEC	PAGE
P(success)=P*	Binomial	1	N	V	30
P(X<x*)=P*	Quantile	1	O	VI	35
E(X)=E(Y)	Sign	2*	O	IX	47
	Mann-Whitney	2	O	VII	40
	Wilcoxon	2*	I	XI	65
E(W)=...=E(Z)	Cochran	≥2*	N	XXIII	108
	Median	≥2	O	XVII	83
	Kruskal-Wallis	≥2	O	XVIII	88
	Friedman	≥2*	O	XXI	99
	Quade	≥2*	I	XXII	104

* indicates that variables must be paired or blocked

TABLE 3

Nonparametric Tests for Independence/Interaction

HYPOTHESIS	NAME	# VARS	LOM	SEC	PAGE
Trend(v. order)	Cox-Stuart	1	O	VII	43
Trend (X v. Y)	Cox-Stuart	2*	O	VII	43
Before/After	McNemar	2*	N	X	51
rho(X,Y)=0	R*C Contingency (chi-square test for indepencence)	2*	N	XI	56
	NP Correlation#	2	O	XIV	69
E(Y|X)=a+b*X	NP Regression	≥2*	I	XV	74
b≠0	Monotone Regress	≥2	I	XVI	79

* indicates that variables must be paired or blocked
# gives pearson's and Spearman's rho,Kendall's Tau

TABLE 4

Nonparametric Tests for Location

HYPOTHESIS	NAME	# VARS	LOM	SEC	PAGE
var(W) = ...=var(Z)	Hartley	>2	I	XX	96
	Squared Rank	>2	I	XIX	92
$F_x(X)=G^*(x)$ (specified parameters)	Kolmogorov (N<31)#	1	I	XXIV	113
	Goodness of Fit# (chisquare,N>14)	1	I	XX	61
$F_x(X)=G^*(x)$ (estimated parameters)	Lilliefors (N<31) (normal,exponential)	1	I	XXV	118
	Goodness of Fit#	1	I	XII	61
	Wilkes-Shapiro (normal,N<31)	1	I	XXVI	121
$F_x(X)=G_x(X)$ F,G: empirical cdfs	Smirnovn	2	O	XXVII	124
	Cramer-von Mises	2	O	XXVIII	127

tests normal,uniform,exponential,weibull,erlang

C. CRITERION II: LEVELS OF MEASUREMENT

Levels of Measurement were treated in section II.

D. CRITERION III: RELATIONSHIP AMONG DATA ELEMENTS

Data set variables may be unrelated, or paired (if 2 variables) or blocked (if more than 2 variables). When unrelated, the order of variables in the first variable is not dependent on the order of of all other variables; e.g., the data elements of a variable could be entered in any order. In data sets with unrelated variables, the number of elements do not have to be equal.

Paired or blocked data sets require sample sizes to be equal and the order of the elements of the variables matters. Examples of pairing might be responses of husbands and wives, the intelligence of twins, the condition of the same experimental subject before and after the experimental treatment.

E. SELECTING THE BEST TEST

As mentioned in section II, two different nonparametric tests applied to the same data set could conceivably give different results to the same hypothesis. The example was the sign test and the Wilcoxon signed rank test. The reason given in section II--that those two tests extract different information from the data--underscores the reason for the decision matrix. In general, the most appropriate nonparametric test is one that tests exactly what is desired, and uses all the available information contained in the data. Using all the available information implies selecting a test which requires the level of measurement of the data, and using a paired or blocked test if appropriate.

These rules require thought. For example, the user might want to know if procedure X produces better results than procedure Y. At first glance, that is a problem of location: is E(X)=E(Y)? However, it could also be considered a problem of distribution: is Fx(z)=Fy(z)? The test for distribution would check the empirical distributions across the entire range of observations, and thus use more of the information of the data set, particularly important with small data sets.

The computer package, described in the previous section, makes running multiple tests on the same data set fairly easy. The user can explore the results of using different tests on the same data, to get insight into his problem.

V. BINOMIAL TEST

A. OVERVIEW

1. Purpose

To characterize p, a population's proportion of "successes", based on the sample.

2. Level of Measurement

Nominal

3. Assumptions

The sample is randomly drawn from the population. Data may be divided into any dicotomous groupings, i. e., two mutually exclusive and collectively exhaustive groups. Here, the groups are called "successes" and "failures".

4. Hypotheses

Let p* be a specified number $0 \le p^* <= 1$.

a. Two tailed test

   (1) Null hypothesis h0: $p=p^*$

   (2) Alternative hypothesis h1: $p \ne p^*$

b. One tailed test

   (1) Null hypothesis h0: $p \le p^*$

   (2) Alternative hypothesis h1: $p > p^*$

c. One tailed test

   (1) Null hypothesis h0: $p \ge p^*$

   (2) Alternative hypothesis h1: $p < p^*$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user specifies which treatment of the data set is used for the test, a hypothesized proportion of successes (p*), and the desired level of significance (alpha*). The

program uses the count of successes and failures; there are three ways to indicate that dichotomization:

a. The user enters cell counts. This is the only way to handle non-numerical data, but will work for any data.

b. The user dichotomizes the data set by a user specified list. This is the way to handle numerical, nominal data. Data values which are elements of the specified list are in one set; the rest of the data values are in the other set. The user indicates whether the list defines success or failure.

c. The data set is divided in two by a partition value. This is only possible if the data is numerical and at least ordinal. All elements above the partition value are in one set; the user defines whether that set is success or failure. The values below the partition are the other type.

2. Program Activation with Cell Counts (choice a above)

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U","V",..."Z" for desired output file. When offered a choice of tests, enter 1.

b. Cell count entry

Lochinvar will ask the number of successes and number of failures--enter those cell counts.

c. Parameter entry

Lochinvar will ask for a hypothesized proportion of successes P*, and a desired level of significance, alpha*. When prompted, enter those numbers

3. Program Activation with Data Values (choice b or c)

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 1. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 1. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index of variable to be used in test. Enter whether dichotomizing by list or by partition value

(2) If by list, enter the number of elements in that list. Enter elements of list. Enter "Y", when prompted if elements on that list represent success.

(3) If by partition value, enter that value. Enter "Y" if data elements below that value are successes.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the data file to which the data has been sent, e.g., "U LISTING". The names of the data set and the variable used in the test are at the top of the output file. The empirical level of significance, alpha-hat, and confidence interval limits, are listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance and Intervals

a. Count number of successes. Divide by the total number of elements to give p-hat.

b. If $N \leq 20$, generate a binomial probability mass function with parameters N and p-hat. Otherwise, use the

normal approximation to the binomial, with parameters N*p-hat and N*p-hat*(1 - p-hat).

c. Compute tail probabilities

(1) The upper tail probability is the likelihood that the number of successes is greater than or equal to the sample number of successes.

(2) The lower tail probabilty is the likelihood that the number of successes is less than or equal to the sample number of successes.

d. Compute the level of signficance, alpha-hat

(1) if h0: p≤p*, alpha-hat equals the upper tail probability.

(2) if h0: p≥p*, alpha-hat equals the lower tail probabiltiy

(3) if h0: p=p*, alpha-hat equals two times the minimum of the upper and lower tail probabilities

e. Compute the interval, N less than 21

(1) Let y1 = number of successes minus one, = index of the pmf of the lower tail probabilty. Vary p*, 0≤p*<=p-hat, until the lower tail probability equals alpha*/2. P* is the lower confidence limit.

(2) Let y2 = number of successes plus one, =index of the pmf of the upper tail probability. Vary p*, p-hat≤p*<=1, until the upper tail probability equals alpha*/2. This p* is the upper confidence limit.

f. Compute the interval, N greater than 20

(1) Compute the standard error of estimate by taking the square root of the quantity (number of successes minus the number of failures, divided by N to the third).

(2) Find the value z, which is the desired quantile, 1-alpha*/2, of the standard normal distribution

(3) The lower confidence limit is p-hat minus z times the standard error. The upper confidence limit is p-hat plus z times the standard error.

2. Program Operations

a. Determine the cell count

b. If N is less than or equal to 20

(1) Generate a vector of N choose k, k=0,1,...,N.

(2) Generate a binomial pmf, with parameters N and p-hat=number of successes divided by N.

(3) Compute the tail probabilities

(4) Compute the empirical level of significance

(5) Using the previously generated N choose k vector, generate partial binomial pmf's, with terms 0 to number of successes minus one, or number of successes plus one to N. A bisection search technique varies p, 0≤p<=p-hat or p-hat<=p<=1, until the sum of the terms of the partial pmf is within 0.0001 of the desired level of significance (alpha*/2, because the interval is two tailed).

c. If the sample contains more than twenty elements

Use the normal approximation to the binomial. The subroutine Muvar uses numerical integration to find tail probabilities; from which the empirical level of significance is estimated. Muvar also funishes the z quantile for computing the confidence interval limits.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400, if the program computes the cell count from a data list; unlimited if the user enters the cell counts directly. Maximum number of variables: 1.

b. Subroutines/Procedures Used

(1) Lochinvar: Binome,Which_vec,Hyp

(2) Crunch: Bintest; Muvar, Cibi21 or Nchuzk, Binpmf, Cibi20, Bisex

(3) External: Shsort

VI. QUANTILE TEST

A. OVERVIEW

1. General Purpose

The quantile test addresses the question of what proportion of the population is less than or equal to a specified value. An example would be that a student might be concerned less with the average grade on a test than the proportion of students who received at least the minimum passing grade.

2. Level of Measurement

Ordinal

3. Assumption

The data set is a random sample from the population.

4. Hypotheses

Here, p* is the specified population proportion, $0 \le p^* <= 1$, and x* is a specified value within the range of the population.

a. Two tailed test

(1) Null hypothesis h0:
the p*th quantile = x*

(2) Alternative hypothesis h1:
the p*th quantile ≠ x*

b. One tailed test

(1) Null hypothesis h0:
the p*th quantile ≥ x*

(2) Alternative hypothesis h1:
the p*th quantile < x*

c. One tailed test

(1) Null hypothesis h0:
the p*th quantile ≤ x*

(2) Alternative hypothesis h1:
the p*th quantile > x*

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify the values for p* and x*, and the letter of the hypothesis desired.

If there is more than one variable in the designated data set, the user must specify which is to be used.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 2. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 2. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index of variables to be used in test, if there is more than one in the data set.

(2) Enter the values of p*, x*, the letter of the chosen hypothesis, and the hypothesized level of significance, alpha*.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the data file to which the data has been sent, e.g., "U LISTING". The names of the

data set and variables used are at the top of the output file. Also included are the test statistic (the number of data elements less than or equal to x*), the level of significance, and confidence limits for the population's p*th quantile.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Calculation of Level of Significance and Confidence Limits

a. Compute the Test Statistic

Crunch compare the data elements with x*, counting the number which are less than or equal to x*. If there are no ties, T1 = T2 = the count. If some elements are equal to x*, then T1 = the count of data elements less than x*, and T2 = T1 plus the number of ties.

b. Computation of the Level of Significance

If T1 = T2, the computation of the level of significance is exactly the same as the binomial test, with T1 as the number of successes, the total number of data elements as n, and p* as the p* of the binomial test. If T1 < T2, then the lower tail probability is calculated using T1; the upper tail probability uses T2; and the computation procedes as with a normal binomial test. See section V for details about computation of level of significance in the binomial test.

c. Find the Confidence Limits

The data elements are ranked; the ranked values are called "order statistics", so that X(1) is the smallest value, X(k) is the largest values in a data set of k values. The confidence limits are found by computing the values r and s, which are indices of the order statistics; once they are found, X(r) is the lower limit, X(s) is the larger limit.

(1) With less than 21 data elements, a binomial pmf is generated, with parameters k = number of data elements, and p = p*. R is found by summing terms of the pmf from the 0th term (P(N=0)) until the first term which causes the sum to be greater than the half of the hypothezed level of significance (alpha*/2). The index of that term is r. S is found by summing terms of the pmf from the kth term (P(N=k)) until the first term which causes the sum to be greater than alpha*/2. The index of that term is S.

This procedure is conservative on both bounds, and therefore may be very conservative overall. For example, with k = 18, p* = 0.5, and alpha*= 0.15, Crunch would choose r of 5 and s of 13; the probability that in the population an x would fall between X(5) and X(13) is 0.904, while the user would have been satisfied with about 0.85. With the binomial tables, the user might choose indices 6 and 13, or 5 and 13, which would have only 0.0481 in one tail but 0.1151 in the other, so that the test would have almost exactly the alpha desired.

(2) With more than 20 data elements (=k), the hypothesized variance is k(p*(1-p*)); its square root is the sample standard deviation. The expected value is kp*. R is the expected value plus the quantity standard deviation times the (alpha*/2) quantile of the standard normal distribution. If r is not an integer, round up. S is the expected value plus the quantity standard devation times the (1-alpha*/2) quantile of the standard normal distribution. Again, round up if necessary to make s an integer.

2. Program Method of Operation

The subroutine celler calculates T1 and T2 by calling the subroutine dicot, which takes the values of the data set and find the number less than and the number less than or equal to the specified x*. The subroutine quantile receives that information, then calls the subroutine bintst to perform the binomial test described above.

Quantl then calls subroutines Nchuzk and Binpmf to generate a binomial pmf to calculate the confidence limit indices r and s, if the total number of values is 20 or less. Otherwise, Quantl calls the subroutine Normv for values of the (alpha*/2) and (1-alpha*/2) quantile of the normal distribution to use in the calculation of r and s. The data values are sorted by the subroutine RNQ, and the X(r) and X(s) values dereferenced as the confidence limits.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Only one variable is used.

b. Subroutines/Procedures Used

(1) Lochinvar: Quantile

(2) Crunch: Quantl, Celler, Dicot, Bintst; Rnq, Nchuzk, Binpmf or Normv

(3) External: Shsort

VII. MANN WHITNEY TEST

A. OVERVIEW

1. General Purpose

With two variables, test whether the populations from which the variables were drawn have the same mean.

2. Level of Measurement

Ordinal

3. Assumptions

a. Each variable is a random sample.

b. There is mutual independence between samples.

4. Hypotheses

a. Two tailed test
   - (1) Null hypothesis h0: $E(X)=E(Y)$
   - (2) Alternative hypothesis h1: $E(X)\neq E(Y)$

b. One tailed test
   - (1) Null hypothesis h0: $E(X)\geq E(Y)$
   - (2) Alternative hypothesis h1: $E(X)<E(Y)$

c. One tailed test
   - (1) Null hypothesis h0: $E(X)\leq E(Y)$
   - (2) Alternative hypothesis h1: $E(X)>E(Y)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide which two of the variables to use in the test if the data set contains more than two. The user specifies the level of significance.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 4. If data needs to be entered, enter "N" for first question, "Y" for second; when data entered, choose to do a test; when offered a choice of tests, type 4. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test.

(2) Enter the level of significance.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The empirical level of significance and whether to reject the null hypothesis are stated.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Sums of the Ranks of the First Variable

Rank the values of the two variables. Sum the ranks of each. The test statistic T is the sum of the ranks of the first variable.

b. Overall Test Statistic

The overall test statistic, if their are many ties, is T1. Let n be the number of elements in the first variable, m the number of elements in the second variable; N

is the sum of n and m. Sum the squared ranks of both variables to find R2. The numerator of the test statistic is T minus n(N+1)/2. The denominator is the square root of the difference of two terms: R2*nm/N(N-1), and (nm(N+1)**2)/4(N-1).

2. Program Method of Operation

The subroutine Manwht calls the nonimsl subroutine Utest, to get the level of significance.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

(1) Lochinvar: Manwhit
(2) Crunch: Manwht
(3) External: Utest

VIII. COX-STUART TEST FOR TREND

A. OVERVIEW

1. General Purpose

This is a specialized form of the sign test, to determine if the data elements in the second half of a sample are significantly greater or less than the corresponding values in the first half. With one variable, the order of the data elements determines how the data elements are paired for comparison.

With two variables, in which the data elements are paired, the ranks of the independent variable orders the values of the dependent. This yields a rough test of the interaction or dependence of the two variables.

2. Level of Measurement

Ordinal

3. Assumptions

a. The data elements represent random draws

b. Either there is no trend, or there is an underlying trend upward or downward throughout data.

4. Hypotheses

Where (+) indicates the second data element is greater than the first in the pair, otherwise (-), so that $P(+)=P(-)$ indicates no trend, $P(+)>P(-)$ indicates upward trend (1 variable) or vary directly (1st variable to 2d variable), and $P(+)<P(-)$ indicates downward trend or vary an inverse relationship.

a. Two tailed test

(1) Null hypothesis h0: $P(+)=P(-)$

(2) Alternative hypothesis h1: $P(+)\neq P(-)$

b. One tailed test

(1) Null hypothesis h0: $P(+)\leq P(-)$

(2) Alternative hypothesis h1: P(+)>P(-)

c. One tailed test

(1) Null hypothesis h0: P(+)≥P(-)

(2) Alternative hypothesis h1: P(+)<P(-)

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If one variable is used, make sure data elements are in the order (often order of time) for the trend to be tested. If two variables are to be used, data elements must be paired. The user selects the type of hypothesis and the level of significance. If there are two variables, the user specifies which is dependent.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 3. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 3. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index or indices of variables to be used in test. Enter pairs to be excluded, if using two variables.

(2) Enter letter of hypothesis type and level of significance.

(2) If there are two variables, enter which is dependent.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance

a. Form Pairs (X,Y) Significance

(1) If there is one variable, call it Y, and go to step (2). If two variables, let X be the independent variable. and Y be dependent. Find the order of the X. Use that ordering to re-order Y in ascending order of the values of X.

(2) Let k be the number of elements in the variable Y. If K is even then there will be (k/2) pairs, if odd then ((k-1)/2) pairs. Let np equal the number of pairs.

b. Find Test Statistic

(1) For i=1 to np, compare Y(i) to Y(np-i+1). If Y(i) > Y(np-i+1) then increment the counter for failures. If Y(i) < Y(np-i+1) then increment the counter for successes. The test statistic is from the binomial test, with n equal the number of successes plus failures, p* equal 0.5, and the number of success equal number of times the second half data elements are strictly greater than the corresponding first half elements.

2. Program Method of Operation

a. Ordering the Dependent Variable

The subroutine Coxstu dereferences the X values into the INDEP vector, sorts those values using Shsort, and

uses the indices returned in the vector KEY to dereference the Y values into a vector TLIST.

If there is only one variable, use a do for loop to load its values, in their current order, into TLIST.

b. Computing the Test Statistic

The subroutine Coxstu calls the subroutine Signt to perform the sign test. See section IX.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

(1) Lochinvar: Coxstu
(2) Crunch: Coxstu, Signt
(3) External: Shsort

IX. SIGN TEST

A. OVERVIEW

1. General Purpose

With pairs of data (Xi,Yi), determine if the expected value of the first element, E(X), is greater than the expected value of the second element, E(Y).

2. Level of Measurement

Ordinal within each pair; that is, it can be determined that Xi is greater than Yi (+), or that Yi is greater than Xi (-), or that they are equal. The values of the pair do not have to be ordered with any values outside the pair.

3. Assumptions

a. Random Sample

X,Y are a random sample of the population pairs.

b. Level of Measurement

Ordinal within each pair, nominal among pairs.

c. Invariance of Probability that Xi > Yi

The underlying population from which the sample is drawn is assumed to have a constant $P(X>Y)$.

4. Hypothesis

Here, (+) is the case that $X>Y$, (-) is the case that $Y>X$, where X,Y refer to population random variables.

a. Two tailed test

(1) Null hypothesis h0: $P(+)=P(-)$

(2) Alternative hypothesis h1: $P(+)\neq P(-)$

b. One tailed test

(1) Null hypothesis h0: $P(+)\leq P(-)$

(2) Alternative hypothesis h1: $P(+)>P(-)$

c. One tailed test

(1) Null hypothesis h0: $P(+)\geq P(-)$

(2) Alternative hypothesis h1: $P(+)<P(-)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

It is possible that the data is non-numerical, but ordinal within pairs, e.g., a list of winners of arm-wrestling contest between two platoons as a measure of arm strength. If so, then perform the test as a binomial test, entering cell counts, with number of successes as the number of times Xi > Yi, n is the total number of untied pairs, and p* as 0.5.

With numerical data, the program will make the comparisons. The user must enter the hypothesis letter and the hypothesized level of significance.

If there are more than two variables in the designated data set, the user must choose which are included in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 5. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 5. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter the letter of the hypothesis type and the hypothesized level of significance, alpha*.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic and the level of significance are printed, as well as whether to reject the null hypothesis.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance

a. Calculation of the Test Statistic

For each pair (Xi,Yi), subtract Yi from Xi. Count the number of times the difference is greater than 0: this is the test statistic. The total number of untied pairs is the count of times the difference is not zero.

b. Calculation of the Level of Significance

The level of significance is calculated by the binomial test, with the number of successes equal to the test statistic, the total sample size equal to the number of untied pairs, and the hypothesized proportion of success p* equal to 0.5.

See section V for details about computation of levels of significance by the binomial test.

2. Program Method of Operation

The subroutine Signt compares the values of the lower indexed variable with the corresponding values of the other variable. Values within epsilon (=0.0001 by default) are considered tied. If not tied, then the program checks whether Xi>Yi or Xi<Yi, and increments the appropriate count, (+) or (-). Signt then calls the subroutine Bintst to compute the level of significance.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of pairs: 200. Required number of variables: 2.

b. Subroutines/Procedures Used

(1) Lochinvar: Signtest

(2) Crunch: Sgnt

(3) External: Shsort

X. MCNEMAR TEST FOR SIGNIFICANCE OF CHANGE

A. OVERVIEW

1. General Purpose

With data in pairs (Xi,Yi), and x and y values each classifiable as successes or failures, to determine if a significant change has occurred.

2. Level of Measurement

Nominal

3. Assumptions

a. The pairs are a random sample.

b. X and Y can be divided into mutually exclusive and collectively exhaustive groups: success and failure.

c. Each of the k pairs can be categorized into one of four cells: success/success (1,1), success/failure (1,0), failure/success (0,1), and failure/failure (0,0).

4. Hypothesis

a. Null hypothesis

H0: P(0,1)=P(1,0) for all i

b. Alternative hypothesis

H1: P(0,1)≠P(1,0) for all i

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If the data is nominal, the user must enter the counts of each of the four categories.

If the data is ordinal or above, the user may enter cell counts, or may specify partition values to dichotomize X and Y. Lochinvar will ask for the partition value for X, and whether successes are less than it. For example, if

17.5 is the partition value for X, and successes are greater than it, any X value greater than 17.5 will be a success; depending on its Y value, the pair will be counted in the success/success (1,1) cell, or success/failure (1,0) cell. After asking for the X partition value, Lochinvar will ask for the Y partition value and whether successes lie above or below.

If there are more than two variables in the designated data set, Lochinvar will ask which two are to be used for the test.

2. Program Activation when Using Cell Counts

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U","V",..."Z" for desired output file. When offered a choice of tests, enter 6.

b. Cell count entry

Lochinvar will ask the cell counts for each of the four categories: success/success, success/failure, failure/success, and failure/failure.

Enter the hypothesized level of significance.

3. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 6. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 6. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of the two variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter the partition value for the lower indexed variable and whether successes are less than that value. Enter the partition value for the variable with the higher index; indicate whether successes are less than that value.

(3) Enter the hypothesized level of significance.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file.

If there are 20 or less pairs in the data set which are classified as (1,0) or (0,1), then the test statistic is T2=number of pairs classified as (1,0); that test statistic and level of siginificance will be listed. Otherwise, the test statistic T1, explained below, and the level of significance will be given.

In either case, the program will compare the empirical level of significance with the hypothesised, and state whether the null hypothesis is rejected.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Level of Significance

a. Computation of Cell Counts

If cell counts were provided by the user, go to step b or c. If data values were used, Crunch examines each

pair (Xi,Yi) to see the relation each data element has to its corresponding partition value. Crunch decides which cell the pair belongs to, (1,1),...(0,0), and increments the count for that cell.

b. If Count for Cells (1,0) and (0,1) less than 21

The test statistic is T2, which equals the count of cell (1,0). The binomial test is invoked to find the level of significance. T2 is the number of successes; the total count of cells (1,0) and (0,1) as n, the total sample size; the hypothesized proportion of success, p* is 0.5. See section V for details of computation.

c. If Count for Cells (1,0) and (0,1) exceeds 20

The test statistic is T1, equal to the squared difference between the cell counts (1,0) and (0,1), divided by their sum. This test statistic is compared to the chi-square distribution with one degree of freedom to find the level of significance.

2. Program Method of Operation

a. Cell Count Calculation

The subroutine Celler compares each of the lower indexed values to the first partition value, then each of the other variable's values to the second partition value. At each comparison, a "1" is assigned to an integer vector if the value is a success, otherwise a "0" is assigned. The resulting integer vector, with the same number of elements as the original data set, is then used to find cell counts. With k pairs, the first through the kth elements of the integer vector are multiplied by two, then added to the corresponding k+1st through 2kth elements. The resulting k element vector has values 0,1,2,3, corresponding to the four cells. The vector is then used to compute cell counts.

b. Calculation of the Test Statistic

Crunch figures if the cell counts of (1,0) and (0,1) total more than 20. If so, T1 is computed, otherwise T2 is used.

c. Calculation of the Level of Significance

If T1 is used, the subroutine mcnemar performs linear interpolation of tabled values of the chi-square distribution with one degree of freedom. If T2 is used, the binomial distribution subroutine Bintst is used to calculate the level of significance.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of pairs: 200 if program computes cell counts; no limit if the user enters cell counts.

b. Subroutines/Procedures Used

(1) Lochinvar: Cel_test, McNemar

(2) Crunch: McNemr, Celler, McCount

(3) External: Shsort

XI. CHI-SQUARE TEST FOR INDEPENDENCE (R*C CONTENGENCY)

A. OVERVIEW

1. General Purpose

Used with a data set composed of paired data elements, to test whether the classification or value of the first element of the pair is statistically independent of the classification or value of the second.

2. Level of Measurement

Nominal

3. Assumptions

a. The pairs is randomly drawn from the population.

b. The first element of each pair can be placed in exactly one of r categories (rows), and the second element of each pair can be placed in exactly one of c categories (columns).

4. Hypothesis

a. Null hypothesis

HO: P(pair is in row i and col j) equals P(first element in row i) times P(second element in row j)

b. Alternative hypothesis

H1: P(pair is in row i and col j) does not equal P(first element in row i) times P(second element in row j)

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

a. Entry of cell counts

If the data is nominal, the user must decide the categories of the rows and columns, and enter the observed cell counts, Oij, for each cell. The user may use this technique for data of a higher level of measurement.

b. Entry of fixed row and column totals

If the data is ordinal or above, and the categorization of the data points is by their order (e.g., not by whether their last digit is odd), the user may enter Ri's and Cj's. The sums of the Ri's and Cj's must exactly equal the number of pairs in the data set.

c. Entry of row and column partition values

If the data is ordinal or above, and the categorization is based on order, the user can specify (r-1) values to divide the data elements into r rows, and (c-1) values to divide the data elements into c columns. For example, a data pair would fall into cell (1,2) if the first element of the pair were less than or equal to the first row partition value, and the second element of the pair were greater than the first column partiton value but less than or equal to the second.

2. Program Activation when Using Cell Counts (choice a)

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U","V",..."Z" for desired output file. When offered a choice of tests, enter 7.

b. Cell count entry

Lochinvar will ask the number of rows and columns. It will then prompt entry of the observed cell count for each cell.

3. Program Activation Using Data Values (choice b or c)

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 7. If data needs to be entered, enter "N" for first question, "V" for second; enter the

data; when offered a choice of tests, type 7. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of the two variables to be used in test. Enter the number of rows and the number of columns. Enter whether using fixed row and column totals or partition values.

(2) If by fixed totals, enter the row totals when prompted, then the column totals.

(3) If by partition value, enter the row partition values when prompted, then the column partition values.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The data set and variable names are at the top of the output file. The test statistic T, the degrees of freedom, and the matrix of observed cell counts are listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Computation of Test Statistic

The program Crunch will:

a. Find the observed cell count

Specify the r rows and c columns, thus creating r*c mutually exclusive cells. Count the number of pairs which fall into the categorization of each cell: that number is the observed cell count, Oij, of cell ij.

b. Find the expected cell count

All the cell counts in row i are summed to find the row total Ri. All the cell counts in column j are summed to find the column total Ci. The expected cell count Eij is found by multiplying Ri times Cj and dividing by the total number of pairs, N.

c. Find the test statistic T

For each cell, subtract Oij from Eij. Square each difference. Divide each squared difference by the corresponding Eij. Sum those quotients to find the test statistic T, which can be compared to a chi-square distribution with (r-1)*(c-1) degrees of freedom. The quantile of the chi-square distribution corresponding to T indicates the empirical confidence coefficient, or the complement of the empirical level of significance.

2. Program Operations

a. Computation of the cell counts

If the user supplied the cell counts, go to step b. If the user gave fixed row and column totals, rank all the first elements to find the row partition values, and rank all the second elements to find the column partition values. Compare the first element of each pair to the row partitions: assign to each pair the index of the largest partition value which the first element is less than; if the first element is larger than the largest partition value, it is assigned the integer r. Do the same with the second elements and the column partition values. Each pair now has two integers assigned, which designate membership in a row and in a column. Count the number of pairs falling in each cell: this is the observed cell count Oij.

b. Compute the test statistic T

Find the row and column totals by summing Oij's. Compute the expected cell counts Eij by Ri*Cj/N. Square the difference (Oij-Eij), divide by Eij, sum over all cells.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400, if the program computes the cell counts; unlimited if the user supplies cell counts. Maximum number of variables/ treatments: 2. Maximum total cells (r*c): 50.

b. Subroutines/Procedures Used

(1) Lochinvar: Cel_test,Rc_cont; or Which_test, Rc_cont,Set_cell,Which_vec, Exclud_b

(2) Crunch: Celler,Partv,Tc3 or Tc4,Rcctr,Rcont

(3) External: Shsort

XII. CHI-SQUARE GOODNESS OF FIT TEST

A. OVERVIEW

1. General Purpose

With a sample of 15 or more data elements, test if that sample could be drawn from a specified distribution.

2. Level of Measurement

Interval

3. Assumptions

The variable contains a random sample.

4. Hypothesis

Where $F^*(x)$ is hypothesised distribution function, and $F(x)$ is the distribution of the population.

a. Null hypothesis

$H0: F(x)=F^*(x)$ for all X

b. Alternative hypothesis

$H1: F(x)\neq F^*(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide among five distributions for which to test. Parameters may be specified by the user, or may be estimated by the program (using method of moments).

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 8. If data needs to be entered, enter "N" for first question, "V" for second; enter data,

and when offered a choice of tests, type 8. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter theoretical distribution . Choose among: uniform, normal, exponential, erlang and wiebull.

(3) Enter the parameters of the theoretical distribution, or request that the program estimate the parameters. In the normal and exponential the method of moments can be invoked to allow the program to estimate the maximum likelihood estimates of the parameters. In the case of the erlang, the user must specify the integer shape parameter n, but can allow the program to estimate the scale parameter lambda. In the case of the uniform and the weibull, the user must specify the parameters

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic T and the degrees of freedom will be displayed. Finally, the matrix of cell counts, expected and observed, will be given.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculation of the Expected Cell Counts

Let k equal the number of data elements. If $15 \le k <= 50$, divide k by 5 and round to the nearest integer.

This will be C, the number of cells; e.g., k=17 implies C=3, k=18 implies C=4. In all cells except perhaps the first and last the expected cell count will be 5. If C*5=k, then the expected cell counts for all cells is 5. If C*5=k+1, the expected cell count of the first cell is 4; if C*5=k+2, the expected cell count of the first and last cells is 4. If C*5=k-1, the expected cell count of the first cell is 6; if C*5=k-2, the expected cell count of the first and last cells is 6. For example, if k=17, C=3; since 3*5=17-2, the first and last cells have expected counts of 6, i.e., the expected cell counts are 6,5,6.

If k>50, there will be 10 cells. Divide k by 10 and truncate; that will be the minimum expected cell count E. The remainder of the truncation is rem=k-10*E; add 1 to the cell count of the last rem cells.

The expected cell counts are stored in the vector EJ. This is the building block of the F=P(X≤x) values of the theoretical distribution. Scan the EJ vector to find the CUM vector, i.e., CUM(1)=EJ(1), CUM(2)=EJ(1)+EJ(2), ..., CUM(j)=sum of EJ(i), i=1,..,j. Divide each element of the CUM vector by the number of elements, k, to form the CUMP vector.

b. The Partition Values

The usual form of theoretical distributions takes a value x and returns a value F, such that F = P(X≤x). In this case, the inverse form of the theoretical distributions is used to take an F, contained in the CUMP vector, and return an x. The first (C-1) values returned form the partition values which will divide those data elements into cells.

c. The Observed Cell Count

The data elements are compared to the partition values contained in the vector PARTV. An element falls in the ith cell if it is larger than PARTV(i-1) but less than

or equal to PARTV(i). Elements less than PARTV(1) are in the first cell; elements larger than PARTV(C-1) are in the last cell. The observed cell counts are contained in the vector RCCT.

d. The Test Statistic

For each cell, subtract EJ(i) from RCCT(i), square that difference, and divide that square by EJ(i). Sum over all cells. This is the test statistic T, with degrees of freedom (C-1).

2. Program Method of Operation

The subroutine Gdfit calls the subroutine Muvar to find the sample mean and variance if parameters must be estimated. The subroutine Cpvgen figure the EJ, CUM, and CUMP vectors. The vector PARTV is figured in one of five subroutines, depending on the theoretical distribution specified: unid, normd, expd, erld, weid. The observed cell counts RCCT are found in Celcnt. The test statistic T is computed in the subroutine Gdfit.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variable: 1.

b. Subroutines/Procedures Used

(1) Lochinvar: Gd_fit_n, Gd_fit_e, Gd_fit_u, Gd_fit_er, Gd_fit_w, Gd_fit

(2) Crunch: Gdfit, Cpvgen, Muvar, Celcnt, Unid, Normd, Expd, Erld, Weid

(3) External: Shsort

XIII. WILCOXON SIGNED RANK TEST

A. OVERVIEW

1. General Purpose

With two paired variables (X,Y), the signed rank test determines whether the expected values of populations X and Y could be the same.

This requires interval data. If the data is only ordinal, use the sign test.

2. Level of Measurement

Interval

3. Assumptions

The pairs (Xi,Yi) are random samples from the population of all pairs.

4. Hypotheses

a. Two tailed test
 - (1) Null hypothesis h0: $E(X)=E(Y)$
 - (2) Alternative hypothesis h1: $E(X)\neq E(Y)$

b. One tailed test
 - (1) Null hypothesis h0: $E(X)\leq E(Y)$
 - (2) Alternative hypothesis h1: $E(X)>E(Y)$

c. One tailed test
 - (1) Null hypothesis h0: $E(X)\geq E(Y)$
 - (2) Alternative hypothesis h1: $E(X)<E(Y)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

To make the test more flexible, a constant term can be added to the higher indexed variable, so that the test becomes a question of the relationship of $E(X)$ and $E(Y)+b$.

By default, the value of b is 0; but the user can specify any real number.

Lochinvar also asks for the hypothesis type and level of significance.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 9. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 9. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any.

(2) Enter whether a constant term is to be added to the value of the higher indexed variable, Y. By default, the constant is 0.

(3) Enter the letter of the type of hypothesis and the hypothesised level of significance.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file.

Crunch will provide the sum of negative ranks and sum of positive ranks, as well as the test statistic T in

all cases. If there are fewer than 20 pairs, table look up is required; otherwise, the program will compute the empirical level of significance.

When the number of untied pairs is less than 20, the output will also contain the vector Dif, the differences Yi-Xi, in ranked order.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Sums of the Negative and of the Positive Ranks

Two sums are calculated, one for the negative differences-- those pairs in which Xi>Yi--and one for the positive differences. Pairs in which Xi=Yi are not considered. Subtract Xi from Yi, to form the vector Dif. Take the absolute value of Dif to form Adif. Rank the values of Adif to form the vector Rank. If Yi<Xi, the ith value of Rank is added to the sum of negative ranks, then that ith value is multiplied by negative 1. If Yj>Xj, the jth value of Rank is added to the sum of positive ranks.

b. Computation of Test Statistic

The vector Rank was originally the ranks of the absolute differences; now, the ranks for the negative differences have been multiplied by -1. Sum the vector Rank to find the numerator of the test statistic. Sum the squares of the elements of Rank to find the denominator of the test statistic. Divide numerator by denominator.

c. Level of Significance

If the number of untied pairs is 20 or less, use table A-13 [ref 1], using the sum of positive ranks. Otherwise, Crunch computes an approximation, using the test statistic T and the standard normal distribution.

2. *Program Method of Operation*

The subroutine Wilcox finds the Dif and Adif vectors, and calls the subroutine Ranque (which in turn calls Rnq) to find the vector Rank. Wilcox computes the sums of the positive and negative ranks, as well as the test statistic T. If the number of untied pairs is 20 or less, it will display the sums of ranks. Otherwise, it will call the subroutine Normv to find the approximate empirical level of significance, and display whether to reject the null hypothesis.

3. *Program Characteristics*

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2; maximum number of pairs: 200.

b. Subroutines/Procedures Used

(1) Lochinvar: Wilcoxon

(2) Crunch: Wilcox,Ranque,Rnq,Normv

(3) External: Shsort

XIV. NONPARAMETRIC CORRELATION

A. OVERVIEW

1. General Purpose

With n pairs of data elements, X and Y, test whether the value of X is statistically independent of the value of Y, or if it varies directly or inversely with Y.

Three different test statistics are available by running this one program: Pearson's rho, Spearman's rho, and Kendall's tau.

2. Level of Measurement

Interval for the Pearson's rho to have meaning.

Ordinal for Spearman's rho and Kendall's tau.

3. Assumptions

X,Y are a random sample of population pairs.

4. Hypothesis

Where rho is a measure of correlation,-1≤rho<=1

a. Null hypothesis

H0: rho(X,Y) = 0, a necessary condition for
statistical independence

b. Alternative hypothesis

H1: rho(X,Y) ≠ 0

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user chooses which pair of variables from a data set are used in a test, and whether to exclude any of the pairs from the test. As described below, the computation of Kendall's tau uses an algorithm of complexity level n-squared (when comparing every pair with every other pair);

if there are more than 40 pairs Lochinvar offers the user the chance to skip the Kendall's tau test. If so, only Spearman's rho and Pearson's rho (which may have no meaning if the data is ordinal) will be computed. Otherwise, all three will be produced.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 10. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 10. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) If there are more than two variables in the data set, enter the indices of the two to be used. Enter pairs to be excluded, if any.

(2) If there are more than forty pairs, enter whether the Kendall's tau should be computed.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. Spearman's rho, Pearson's rho, and Kendall's tau are listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistics

a. Pearson's Rho

Pearson's rho is SXY divided by square root of SXX times SYY. The deviations of X are found by subtracting the mean of X from each value of X; deviations of Y are Yi minus mean of Y. The product of deviation of Xi times deviation of Yi, summed over all pairs, is SXY. The sum of squared deviations of Xi is SXX; the sum of squared deviations of Yi is SYY.

b. Spearman's Rho

Rank the values of X. Rank the values of Y. Those pairs of ranks (R(Xi),R(Yi)), are then used to compute SXY divided by square root of SXX times SYY. That is, Spearman's rho is Pearson's rho performed on the ranks of the values rather than the values themselves.

c. Kendall's Tau

Tau is roughly the measure of the number of concordant pairs minus disconcordant pairs divided by total number of possible pairing. Concordance implies positive correlation; e.g., if pair of the data set number 4 (X4,Y4) has an X value greater than, say, X5, and a Y value greater than Y5, then this is one small indication that there is positive correlation between X and Y. Therefore, the 4th and 5th pairs of the data set form to make one concordant pair. If X6 is greater than X4, but Y6 is less than X4, this is a little evidence of negative correlation; the 4th and 6th pairs of the set form to make one disconcordant pair. Ties in either X or Y mean that the comparison yields neither concordance nor disconcordance.

To compute tau, order the pairs (Xi,Yi) in ascending order of X's. Compare the first pair (the one with the lowest X value) to every other pair. If X1 is tied with

Xi or Y1 with Yi, go to the next pair; if not, (Y1 < Yi) implies concordance, (Y1 > Yi) implies disconcordance; increment the relevant count. After the first pair is compared with all others, the second pair is compared with all succeeding pairs (3,4,...,k), incrementing counts as necessary. Each pair is compared to all succeeding pairs, until Xk-1 is compared to Xk. Kendall's tau equals the number of concordant pairs minus the number of disconcordant pairs, all divided by the total possible number of pairings (k(k-1)/2). If all pairings are concordant, tau will equal 1; all disconcordant pairings will yield an -1.

2. Program Method of Operation

a. Pearson's and Spearman's Rhos

The subroutine Rhoer calculates SXY, SXX, SYY for any input values, and returns--inter alia--rho. For Pearson's rho use the values of the data set. For Spearman's rho, use subroutine Rnq to rank the values of the data set, and send those ranks to Rhoer.

b. Kendall's Tau

Since the values have already been ranked, use the indices of ranked X to order the pairs (R(Xi),R(Yi)). Ranks work the same as values for finding concordance or disconcordance, but ties are easier to discern with ranks. With nested do for loops, compare the ith pair (i = 1 to k-1) with pairs j = i+1 to k; R(Xi) = R(Xj) or R(Yi) = R(Yj) means to continue with looping. Otherwise, if R(Yi) < R(Yj) then increment concordant counter, if R(Yi) > R(Yj) then increment disconcordant counter. At the end of the nested loops, divide the difference of the concordant counter minus the disconcordant counter by the term (k(k-1)/2).

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of pairs: 200.

b. Subroutines/Procedures Used

(1) Lochinvar: Correl8
(2) Crunch: Correl,Rnq,Rhoer
(3) External: Shsort

XV. NONPARAMETRIC REGRESSION

A. OVERVIEW

1. General Purpose

Regression tests the effect of the value of independent variable X has on the value of the independent variable Y. The linear regression model assumes that the effect of X on Y is linear, that is, $E(Y|X)=E(Y)+b1(X-E(X))$. This test:

(1) Finds estimates of b1.

(2) Predicts values of Y given values of X

(3) Test if the linear model is correct

(4) Finds confidence limits for the value of b1

2. Level of Measurement

Ordinal, to test the linear model

Interval, for the other purposes listed above

3. Assumptions

a. Random sampling of pairs. X may be specified, as long as the Y portion of the pair, given X, is independent.

b. The linear model is the correct model.

4. Hypothesis

Where b1 is slope the parameter; b1* hypothesized.

a. Null hypothesis

$H0: b1=b1^*$

b. Alternative hypothesis

$H1: b1 \neq b1^*$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

a. The user chooses the two variables to be used, and designates which one is dependent. The program will

automatically fit the least squares model, finding b1. The intercept parameter b0, E(Y|X)=b0+b1*X, is also computed.

b. The user can input values for X to predict values of Y based on the least squares parameters.

c. The user can specify a hypothesised b1* to test whether a linear model is appropriate. The default value is b1* = 0; this is the classical test of whether an independent variable predicts a dependent variable.

d. The user can request confidence intervals on the slope parameter b1. As described below, the algorithm to compute those intervals is of degree of complexity n squared, that is, for n pairs(x,y), n(n-1)/2 comparisons must be made and stored to find the confidence intervals. For larger values of n, the user should consider not requesting this option.

If the user does choose to have the confidence intervals computed, the value W will be requested by Lochinvar. That value is explained by Conover [Ref. 1: p. 267], and contained in table A-12 of that reference.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 11. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 11. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter pairs to be excluded, if any.

(2) Enter values of X to compute E(Y|X) based on least squares linear fit.

(3) Enter hypothesized slope parameter b1*.

(4) Enter whether confidence limits are desired.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The least squares fit parameters will be listed, and the Spearman's rho statistic for the specified value of b1. If the user requested expected values of Y for some values of X, the values of X and Y will be listed. Finally, if confidence bounds were requested, those will be given.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Find the Regression Parameters b1,b0

This step is identical to parametric simple regression. Find the means of X and Y, and the vectors of deviations (Xi-mean of X), (Yi-mean of Y). Multiply the corresponding deviations of X and Y together, and sum the products to form SXY. Square the deviations of X, then sum those squares, to form SXX. Sum the squared deviations of Y to find SYY. The slope parameter b1 is SXY/SXX. The intercept parameter b0 is the mean of Y minus b1 times the mean of X. The expected value of Y given X is b0 plus b1 times X. Residuals are the difference between the expected and actual value of Y.

b. Test the Value of X as a Predictor of Y

In parametric regression, it is standard to test whether the slope parameter b1 could be 0 in the entire population. In this package, any value b1* can be set as the null hypothesis. The test checks to see if the residuals of the fit using b1* are correlated with the values of X; if not, then the fit is considered adequate. The residuals are (Yi-b1*Xi), the vector of residuals and the independent variable X are compaired using the Spearman's rho test. See section XIV for details of that test.

c. Predictions of Y based on Least Squares Fit

With the least squares parameters b0 and b1 already computed, Crunch multiplies each given X times b1, adds b0, and displays the predicted Y.

d. Confidence Bounds for the Slope Parameter b1

Form the vector of slopes S. S(1) is (X(1)-X(2)) divided by (Y(1)-Y(2)); S(2) is (X(1)-X(3))/(Y(1)-Y(3)), and so on. With k pairs (X,Y), there are N=k(k-1)/2 elements of the slope vector, when comparing each i with each other j once.

Sort the vector S. The indices of the confidence limits are r (0.5*(N-W)) and s (0.5*(N+W)+1), where N is the total number of slopes and W is the user entered value from table A-12 of Conover. If necessary, round r or s up to integers. The lower limit on b1 is S(r), the upper limit is S(s).

2. Program Method of Operation

The subroutine Regret calls the subroutine Rhoer to find the least squares parameters b1 and b0. Regret figures the residuals for the specified b1*, and sends the residuals to subroutine Correl to have Spearman's rho calculated. Regret figures the vector of slopes S; that vector is sorted by Shsort; Regret finds indices s and r and the confidence limits to b1.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

(1) Lochinvar: Regret

(2) Crunch: Regret, Rhoer, Correl

(3) External: Shsort

XVI. MONOTONIC REGRESSION

A. OVERVIEW

1. General Purpose

With independent variable X, and dependent variable Y, find the E(Y|X) equation, based on a least squares fit of the ranks of X and Y. Linear interpolation finds values from X to E(rank of X) to E(rank of Y) to E(Y).

2. Level of Measurement

Interval

3. Assumptions

a. The sample is a random sample

b. The regression of Y on X is monotonic.

4. Hypothesis

This test does not produce a test statistic; the assumption is that the monotonic model is correct. Instead, it will predict expected values of Y for given values of X, Also, it can produce a trace of the monotone regression, which is analogous to the straight line of the least squares fit of linear regression.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify the index for the independent value X, and dependent variable Y. The user can include values of X to have predicted a corresponding value of Y. The user can request a trace of the monotone regression.

2. Program Activation

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 12. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 12. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables X and Y. Enter pairs to be excluded, if any.

(2) Enter whether to predict values of Y for specified values of X. If yes, enter values of X.

(3) Enter whether a trace is desired.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. In all cases, there will be listed the parameters b0 and b1, the intercept and slope of the least squares fit on the ranks of X and Y. If the user specified X's to predict Y's, the predicted values will be listed. Finally, if a trace was requested, the pairs of X's and Y's will be given.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculations

a. Least Squares Fit on Ranks of X and Y

Sort and rank X and Y. With n pairs, X(i) and Y(j) are the order statistics, RX(i) and RY(j) are the ranks, i=1,...,n, h=1,...,n. Compute the least squares fit

to find b0 and b1, such that E(RY|RX)=b0+b1*RX. See section XV for details of least square computations.

b. Predication of Y Based on Values of X

Linear regression will extrapolate values of Y for a given value of X if that X is outside the range of the observed values X(1), ..., X(n). Monotone regression will not extrapolate; if the user specifies an X outside the range of the observations, that X will not yield an expected Y.

Let Xp be one of the values the user specified to predict a Y, Yp. Find the pair of values of the order statistics of X which bracket Xp, X(i)≤Xp<=X(i+1). The expected rank of Xp, RXp, must fall between those order statistics' ranks, RX(i)≤RXp<=RX(i+1). Use linear interpolation to find RXp: RXp= RX(i) (Xp-X(i))/(X(i+1)-X(i)) *(RX(i+1)-RX(i)).

Use the parameters b0 and b1 to find RYp, the rank of Yp: RYp=b0+b1*RXp.

Find the ranks of Y which bracket RYp, RY(j)≤RYp<=RY(j+1). The value of Yp lies between Y(j) and Y(j+1). Use linear interpolation to find Yp: Yp= Y(j) + (RYp-RY(j))/(RY(j+1)-RY(j)) *(Y(j+1)-Y(j)).

c. Trace of Monotone Regression

The trace is a set of lines, between a sequence of points (Xi,Yi), which describe the monotone regression. When plotted, the trace can be used to predict a value of Y for an X. This is similar to using the straight line of linear regression, Y=b0 + b1*X, for prediction.

To find the endpoints of the trace, use X(1) and X(n), and the procedure of part b, to find E(Y|X(1)) and E(Y|X(n)).

Intermediate points of the trace will use the values of Y to predict expected X. Each value of Y has a rank, RY, which is used with b0 and b1 to find RXt (trace),

RXt= (RY-b0)/b1. For each RXt, find the pair of RX's which bracket it. Use linear interpolation to find Xt.

2. Program Method of Operation

The subroutine Monore calls Rnq to rank and sort values. The subroutine Correl finds the least squares parameters b0 and b1 from the rank vectors RX and RY. The subroutine ESTY performs estimations of Y for given X's, calling Rtov (which calls Interp) to do the linear interpolations. The subroutine Trace performs the trace, calling Vtor, Rtov and Interp.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

(1) Lochinvar: Monoreg

(2) Crunch: Monore, Rtov, Vtor, Esty, Interp, Rnq

(3) External: Shsort

XXVII. MEDIAN TEST

A. OVERVIEW

1. General Purpose

With two or more samples, this test will determine if all the populations from which the samples were drawn could have the same median.

As described below, the package allows combining of data set variables into composite variables in order to test population groups for common median.

The median may be a surrogate for expected value.

2. Level of Measurement

Ordinal

3. Assumptions

The variable were randomly drawn from the respective populations, and are mutually independent.

4. Hypothesis

With k populations, X(1),X(2),...,X(k), and the median denoted as M,

a. Null hypothesis

$H0: M(X1)=M(X2)=...=M(Xk)$

b. Alternative hypothesis

$H1: M(Xi) \neq M(Xj)$ for at least one of the pairs

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user chooses which variables of the data set to include in the test.

Aggregate variables may be formed by having Crunch combine the expected cell counts of several variables into

one or more aggregate variables. For example, let data set Farmland contain six variable, in order, A through F. Variable A is corn yield per acre of farm land treated with fertilizer and pesticide. B is corn yield with fertilizer but no pesticide. C is corn yield with pesticide but no fertilizer. D is corn yield per acre from land untreated by either fertilizer or pesticide. Variables E and F have other values. Variables A and C could be combined into aggregate variable 1, and B and D into 2, to test whether the median yield with pesticide is equal to the median yield without pesticide. Similarly, A and B, and C and D, could be combined into aggregate variables 1 and 2, to test the median of yields with and without fertilizer.

Proper planning and labeling of the variables will facility this aggregation.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 13. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 13. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter indices of variables to be used in test. Enter blocks or pairs to be excluded, if any. Enter whether to use aggregate variables.

(1) If aggregate variables are not used, enter indices of variables to be used in the test.

(2) If aggregates are used, Lochinvar will prompt for a choice on each data set variable, i=1 to the number of variables in the data set. If the ith data set variable is to be excluded from the test, enter 0. If it is to be included, enter the index of the aggregate variable into which it should be combined. For example, with the Farmland data set described above, Lochinvar would prompt user choices on data set variable i= 1 to 6. Let the test be a comparison of yields with or without pesticide, thus variables A and C into aggregate variable 1, B and D into 2, and E and F excluded. When prompted for the data set var 1, enter 1. For data set var 2, enter 2. For data set var 3, enter 1. For data set var 4, enter 2. For data set var 5 and 6, enter 0.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The program will print the test statistic and the degrees of freedom, as well as a two by number of variables contingency table, showing how many elements of each variable are less than or equal to the grand median of all variables, and how many are greater than.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculation of the Cell Counts of the M Matrix

The M matrix has two rows and a column for each variable in the test. The A row has the number of elements in each variable less than or equal to the grand median; the

B row has the number of elements greater than the grand median. The values of all the variables are copied into a vector, which is sorted; the median of all those values is the grand median. The counts of the cells of the matrix M are set to 0. Each element of the first variable is compared to the grand median; if less than or equal, the A row first column cell count is incremented, otherwise increment B row first column cell count. After all elements of the first variable are compared to the grand mean, do the second variable, incrementing the 2 column's cell count, et cetera.

b. Computing the Test Statistic

Sum all counts of the A row to find Atot, and all counts of B to find Btot. Sum Atot and Btot to find N. Sum the first column counts to find NAV(1), the second column counts to find NAV(2), et cetera. The expected cell count for cell aj is Atot*NAV(j)/N; for bj, Btot*NAV(j)/N. The test is in essence the chi-square test for independence (r*c contingency table), with the test statistic as the sum over each cell of observed minus expected count squared, divided by the cell expected count. The actual equations used exploit the structure of the problem. The degrees of freedom are (r-1)(c-1) = number of variables minus one.

2. Program Method of Operation

a. Aggregation of Variable

The subroutine Media assumes that aggregate variables are always used. When not, the program uses k aggregate variables where k equals the number of data set variables included in the test; aggregate variable 1 is data set variable 1,...,aggregate variable k is data set variable k. When aggregates are in fact specified, their values, contained in rp vector, will be used. In either case, the vector AGVARI will contain the indices relating each data set variable to an aggregate variable.

b. Cell counts

The matrix M is a fiction. The vector NA is the count of elements less than or equal to the grand median; it is arranged according to aggregate variable indices. NB is the count of elements greater than the grand median. In a do for loop, i=1 to the number of data set variables, k is set to AGVARI(i), and each element of the ith data set variable is compared to the grand median. If the element is less than or equal to the grand median, increment NA(k) (=NA(AGVARI(i)), otherwise, increment NB(k).

c. Compute the Test Statistic

NA(i) and NB(i) are summed to find NAV(i). All elements of NA are summed to find A, all elements of NB are summed to find B. A plus B equals N. The test statistic is computed using the specialized equations of Conover [ref 1, p. 172]. The degrees of freedom are the number of aggregate variables minus one.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables or aggregate variables: 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Median, Ag_med,Which_vec,None_ex

(2) Crunch: Media

(3) External: Shsort

XVIII. KRUSKAL WALLIS TEST

A. OVERVIEW

1. General Purpose

With a data set of more than two variables, X1,X2,...,Xk, test whether the expected values of all variables are equal.

If this test's null hypothesis is rejected, then it is possible to determine which of the pairs of variables have an unequal mean.

If the data set is blocked, consider instead of this test the Cochran (nominal), Friedman (ordinal) or Quade (interval) tests. If paired, consider the Sign (ordinal) or Wilcoxon (interval) tests.

2. Level of Measurement

Ordinal

3. Assumptions

a. Each variable is a random sample from within its population.

b. The sampling was mutually independent among the populations.

c. The population distributions are identical, or else some of the populations tend to have higher values than others.

4. Hypothesis

a. Null hypothesis

H0: E(X1)=E(X2)=...E(Xk)

b. Alternative hypothesis

H1: E(Xi)≠E(Xj) for at least one pair of populations represented by the data set.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify which variables of a data set are included in the test.

After the test is run, Crunch will present the overall test statistic, and a series of comparisons between the variables. If, and only if, the null hypothesis has been rejected by the overall test statistic may the user infer any information by the comparisons between the pairs (Xi,Xj) of variables.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 14. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 14. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar, enter indices of variables to be used in test.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The overall test statistic will be given. Also listed will be comparisons of pairs of variables; these

comparisons only have meaning if the null hypothesis has been rejected. The paired comparison are essentially a t test run on the ranks of the two variables being compared, testing the hypothesis that their mean ranks are the same. Crunch supplies the absolute difference in their mean ranks and the pooled standard deviation; the user must find the appropriate value of t to conduct the t test.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Calculation of Test Statistic for the Null Hypothesis

a. Find the Sums of Variable Ranks Ri

Rank all values in the data set. Sum the ranks in the ith variable to get the value Ri. The number of data elements in the ith variable is ni, so the average rank in the ith variable is Ri/ni. The sum of ni over all k variables is N.

b. Find the Test Statistic

The test statistic numerator is the sum over k treatments of (Ri**2)/ni, minus ((N+1)**2)*N/4. To find the test statistic denominator, S2, first sum over all data elements the square of the ranks. Subtract from this sum of squares the term ((N+1)**2)*N/4). Divide that difference by (N-1) to get S2. Divide the numerator by S2 to find the test statistic T.

c. Make Comparisons of the Pairs of Variables

As mentioned above, this step is the student t test on the ranks of two variables to see if their population mean could be equal; it means nothing if the null hypothesis has not been rejected.

For each pair of variables Xi,Xj, find the absolute difference in mean ranks, |Ri/ni-Rj/nj|. Find the pooled variance, (S2*(N-1-T)/(N-k)) times (1/ni+1/nj). The

pooled standard deviation is the square root of the pooled variance. If the absolute difference is greater than the pooled standard deviation times the desired value of t, reject the assertion that the mean of populations i and j are the same.

2. Program Method of Operation

The subroutine Kruwal calls Ranque to rank the values of the data set. In a nested do for loop, i=1 to k, j=1 to ni, the intermediate terms of the statitics are computed. Kruwal prints the overall test statistic to the specified output file. It then makes pairwise listings of the absolute difference of mean ranks and the corresponding pooled standard deviations.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Kruwal

(2) Crunch: Kruwal, Ranque

(3) External: Shsort

XIX. SQUARED RANK TEST FOR EQUAL VARIANCE

A. OVERVIEW

1. General Purpose

With two or more treatments in a data set, test whether the population variances of those treatments are equal.

2. Level of Measurement

Interval

3. Assumptions:

Random sampling within each treatment population, and among treatment populations. Level of measurement is interval.

4. Hypotheses

a. Two tailed test, where X,Y are treatments

(1) Null hypothesis h0: $var(X)=var(Y)$

(2) Alternative hypothesis h1: $var(X)\neq var(Y)$

b. One tailed test, two treatments

(1) Null hypothesis h0: $var(X)>=var(Y)$

(2) Alternative hypothesis h1: $var(X)<var(Y)$

c. One tailed test, two treatments

(1) Null hypothesis h0: $var(X)=<var(Y)$

(2) Alternative hypothesis h1: $var(X)>var(Y)$

d. Number of treatments (Xi) are three or more

(1) Null hypothesis h0: $var(X1)=var(X2)=...=var(Xk)$

(2) Alternative hypothesis h1: $var(Xi)\neq var(Xj)$, for some i, j, indices of the data set

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The program allows the user to choose which treatments to include in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 15. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 15. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar, Enter indices of variables to be used in test.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables appear on the top of the output file. Also included are the test statistic and the number of treatments used to compute the statistic.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic (T1/T2)

a. Find the ranks of each absolute deviation:

Compute the sample mean for each of k treatments. Subtract the treatment mean from each of the treatment's elements. Take absolute values of each of those differences. Rank all N absolute differences (N=total elements in data set).

b. Complete intermediate statistics:

(1) SUMR, the sum of all ranks.

(2) SUMR2, the sum of the square of all ranks.

(3) SUMR4, the sum of all the fourth power of each rank.

(4) SUMRJ, the k element vector with the sum of the ranks for each treatment.

(5) SUMR2J, the k element vector with the sum of squared ranks for each treatment.

(6) R2MU, the average squared rank: SUMR2/N.

(7) RMU, the average rank: SUMR/N.

c. For two treatments compute the statistic T1

(1) Numerator: (SUMR2J(1)-n1*R2MU)

(2) Denominatior: SUMR4*n1*n2/(N*(N-1)), minus (RMU**2)*n1*n2/(N-1)

(3) T1=numerator/denominator

d. For three or more treatments compute T2

(1) Numerator: difference between these two terms, sum over k treatments of (SUMR2J**2)/ni, minus (R2MU**2)*N

(2) Denominator: (SUMR4-N*(R2MU**2))/(N-1)

(3) T2=numerator/denominator

2. Program Operations

The program uses subroutine MUVAR to find treatment means. The subroutine Squark finds the absolute deviations;

while the subroutine Rnq ranks them. The subroutine Squark Uses nest "do for" loops, i=1 to k (treatments) and j=1 to ni (elements in treatment), to caluculate the vectors SUMRJ and SUMR2J, and the sums SUMR, SUMR2, SUMR4. RMU and R2MU are found by dividing SUMR and SUMR2 by N. Depending on the number of treatments, T1 or T2 is found by simple arithmetic using the intermediate statistics.

3. Program Characteristics

a. Limitations

Maximum number of data elements is 400. Maximum number of treatments is 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Squarank, which_vec

(2) Crunch: Squark,Muvar,Rnq

XX. HARTLEY TEST FOR EQUAL VARIANCE

A. OVERVIEW

1. General Purpose

Test for equal variance among the k underlying populations of the variables or treatments, reflected by the values for those treatments in the data set.

2. Level of Measurement

Interval

3. Asssumptions

Random sampling within each treatment, independence of sampling among treatments. Interval level of measurement

4. Hypothesis

a. Null hypothesis

H0: var(X1)=var(X2)=...=var(Xk), where k= number of treatments

b. Alternative hypothesis

H1: var(Xi)≠var(Xj) for at least one pair

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The program allows the user to choose which treatments to include in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 16. If data needs to be entered, enter "N" for first question, "Y" for second; when data

entered, choose to do a test; when offered a choice of tests, type 16. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar, Enter indices of variables to be used in test.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The there are the maximum and minimum variances and indices of the associated treatments. Finally, there is the F statistic, to be compared to a table prepared for the Hartley test.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

Crunch does the following: Compute the sample variance for each treatment. Find the smallest and largest variance. Divide the largest variance by the smallest variance to yield the test statistic F. Compare the test statistic to a special table to see if the null hypothesis should be rejected.

2. Program Method of Operation

The treatment variances are figured in the subroutine MUVAR. The subroutine HARTLE scans the variances to find the largest and the smallest, divides the former by the latter, and gives that number as the test statistic F.

3. Program Characteristics

a. Limitations

The maximum number of data elements is 400; the maximum number of treatments is 15.

b. Subroutine/Procedures Involved

(1) Lochinvar: Hartley, Which_vec

(2) Crunch: Hartle,Muvar

(3) External: Sh sort

XXI. FRIEDMAN TEST AND DURBIN TEST

A. OVERVIEW

1. General Purpose

To perform a treatment and block test, similar to ANOVA, using only ordinal data. In fact, the data may be rankable only within each block; i.e., no relationship between data values in different blocks is necessary.

The Friedman test is for data sets of complete block design, i.e., each treatment is applied to each block.

The Durbin test is a generalized Friedman test, for use with incomplete block design. With t treatments and b blocks, each treatment is applied to r blocks (r<b), and each block receives k treatments (k<t). The user must guarantee that the design is balanced, and insert a filler value in place of missing data values when entering the data set.

Each test will generate a set of differences between pairs of treatment summed ranks. If, and only if, the null hypothesis is rejected, these differences may be used, in conjunction with a standard deviation value and the student's t distribution, to determine if there is a significant different between two treatments.

2. Level of Measurement

Ordinal within each block.

3. Assumptions

Values within blocks are independent among blocks.

4. Hypothesis

a. Null hypothesis

HO: Each possible ranking within the block is equally likely (that is, the treatment effects are equal)

b. Alternative hypothesis

H1: At least one of the treatments tends to get higher ranks than at least one other treatment.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide which treatments and blocks to include in the test. If a incomplete block design is to be used, the user must specifically enter the data set for this purpose: there will be t variables, and b elements entered for each variable. In those elements which have a no data value (a treatment which was not applied to some block), enter a filler value. In a data set, there can be only one filler value, and it must be different than any of the data values. For example, if all the data values are positive, -1 could be used as a filler.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 17. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 17. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter whether to use the Friedman or Durbin test.

(2) If the Friedman test is chosen, enter the treatments to be used and any blocks to be excluded.

(2) If the Durbin test is chosen, enter the treatments to be used, any blocks to be excluded, k= number of treatments applied to each block, r= number of blocks to which each treatment is applied, and the filler value.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The overall test statistic for the null hypothesis will be given. In addition, the difference between pairs of treatments' summed ranks will be listed; these are meaningful only if the null hypothesis is rejected.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Use of Filler, If Durbin Test

If the Durbin test is indicated, find the maximum value in the data set. Add 1 to that maximum value to form MP1. In a do for loop, substitute MP1 for the filler value, and create the integer vector INTEST. INTEST(i) equals 1 if the ith element in the data set is not the filler value, 0 if the ith element is a filler.

b. Rank Values Block by Block

In both tests, find the ranks of the values of the data set block by block. Load those ranks into the R vector.

In the Durbin test, the filler elements will have a higher rank than those of data elements, since the fillers were loaded with MP1. Therefore, after all the ranking is complete, multiply the rank vector times INTEST; this sets the ranks of all filler elements to 0.

c. Sums of Treatment Ranks and Total Sums

Square each rank of the rank vector R, and sum those squares over all treatments and blocks to form A2. Sum the ranks of each treatment to form the vector RJ; RJ(j) contains the sum of ranks of the jth treatment.

d. Test Statistic for the Friedman Test

Square each element of the RJ matrix; sum those squares; divide that sum of squares by the number of blocks to find B2. The denominator of the test statistic is A2-B2. Let b be the number of blocks, t the number of treatments. The numerator of the test statistic is (b-1) * (B2 -0.25bk* (k+1)**2). Divide the numerator by the denominator to get the test statistic T2. Compare T2 to the F distribution with (k-1) and (k-1)(b-1) degrees of freedom

e. Test Statistic for the Durbin Test

Sum the squares of RJ. Multiply that sum of squares time 12(t-1). Divide that product by rt(k-1)(k+1), where r is the number of blocks to which each treatment is applied, and k is the number of treatments applied to each block. This quotient is the first term of the test statistic. The second term is 3r(t-1)(k+1) divided by (k-1). Subtract the second term from the first term to find the test statistic T. T is compared to the chi-square distribution with t-1 degrees of freedom.

f. Paired Comparisons of Treatments

Crunch will print the following numbers in all cases, but they only have meaning when the null hypothesis is rejected. For each i,j of the treatments, the absolute difference between the sums of treatment ranks is listed. The number that is equivalent to the standard deviation is also computed. For absolute differences greater than the standard deviation times the t quantile (1-alpha/2), the treatments can be considered to have a different effect.

2. Program Method of Operation

The subroutine Friedm performs most of the computations. The subroutine Ranque ranks the values block by block. The subroutine Parcom does the comparisons of treatment sums of ranks.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Friedman

(2) Crunch: Friedm, Ranque, Rnq, Parcom

(3) External: Shsort

XXII. QUADE TEST

A. OVERVIEW

1. General Purpose

With interval data organized into t treatments and b blocks, test whether all treatments have the same effect. If not, treatments can be compared for differences in pairs.

2. Level of Measurement

Interval

3. Assumptions

a. The values of each blocks are independent of the values of the other blocks.

b. The sampling within each block can yield a range of the elements; that range must be rankable among blocks

4. Hypothesis

a. Null hypothesis

H0: Each possible ranking within the block is equally likely (that is, the treatment effects are equal)

b. Alternative hypothesis

H1: At least one of the treatments tends to get higher ranks than at least one other treatment.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must specify the treatments to be included in the test, and whether any blocks are to be excluded. Crunch will provide the absolute difference between pairs of treatments; if and only if the null hypothesis is rejected, these comparisons can be used to test whether the effects of the pairs are equal.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 18. If data needs to be entered, enter "N" for first question, "V" for second; enter the data; when offered a choice of tests, type 18. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar: Enter indices of variables to be used in test.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic T and its degrees of freedom are listed. Finally, the absolute difference between the pairs of sums of treatment ranks are given. These differences, when used with a standard deviation computed by Crunch and the t distribution, can be used to check for equal effect between pairs of treatments.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Rankings within and among Blocks

Rank each element in the block; assign the elements to the matrix of ranks R. Subtract the largest element from the smallest element; that becomes the block's

range, stored in the vector R. After all blocks have had their elements ranked, rank the vector R; those ranks form the vector Q.

b. The Size Matrix S

With t equal to the number of treatments, the average rank of R is (t+1)/2. Subtract that average rank from each element of R. Multiply the reduced ranks of the first block by Q(1), the reduced ranks of the second block by Q(2), et cetera, to form the matrix S.

c. Sums of the Treatment Elements of S

Sum the elements of the jth treatment of S over all blocks to form the jth element of the vector SJ. Square each element of SJ, sum those squares over all treatment, and divide by b, the number of blocks, to find B1. Square all elements of S, and sum all those squares to find A1.

d. Test Statistic

The test statistic T1 is (b-1) time B1 divided by the quantity A1 minus B1. Compare T1 to an F statistic with degrees of freedom (t-1) and (b-1)(t-1).

e. Pairwise Comparisons

These values have meaning only if the null hypothesis is rejected. Compute the absolute difference between each pair of the vector SJ. If that difference is greater than the student's t quantile times the computed standard deviation, the pair of treatments can be said to have a different effect. The standard deviation is found by taking the square root of (2b(A1-B1)/(b-1)(t-1)).

2. Program Method of Operation

The subroutine Quade calls Ranque to find the matrix R and the vector Q. The test statistic is computed in Quade, as is the standard deviation. The subroutine Parcom makes the pairwise comparison.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Quade

(2) Crunch: Quade, Ranque, Rnq, Parcom

(3) External: Shsort

XXIII. COCHRAN TEST FOR RELATED OBSERVATIONS

A. OVERVIEW

1. General Purpose

With a data set of b blocks, to which k treatments have been applied, test whether the treatments have been equally effective. "Effective" must be reducible to two categories, mutually exclusive and collectively exhuastive. Here, these categories will called success and failure.

If the data is not reducible in this fashion, the Friedman (ordinal) or Quade (interval) tests are available.

2. Level of Measurement

Nominal

3. Assumptions

The blocks are a random sample from among all blocks

4. Hypothesis

a. Null hypothesis

H0: The treatments are equally effective

b. Alternative hypothesis

H1: There is a difference in effectiveness among the groups.

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must decide the standard for success or failure of each treatment. If the data is nominal, the user should enter cell counts. See paragraph 2, below. If the data is ordinal, the user may enter a partition value for each treatment of a data set; see paragraph 3. If this latter method is used, Crunch will classify each value of an

entered data set as a success or failure by its relationship to the partition value for its treatment.

If partition values are used in conjunction with a data set, the user may specify which variables of the data set are to be included in the test, and may designate blocks to be excluded.

2. Program Activation when Using Cell Counts

a. Initiation and test selection

In CMS, type LOCHI. Enter "N" for no to question about previously entered data. Enter "C" for cell counts. Enter "U","V",..."Z" for desired output file. When offered a choice of tests, enter 19.

b. Cell count entry

Lochinvar will ask how many treatments and how many blocks are included in the test. After those numbers are entered, Lochinvar will ask whether block 1, treatment 1 was a success. Enter yes or no. The program will prompt for each treatment in block 1, then move to block 2, and so forth, through the b*k choices. The matrix of 1's (successes) and 0's (failures) will be displayed, and the user has the chance to correct the values for any block (the program will prompt for entries of each treatment of a specified block).

3. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 19. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 19. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

Lochinvar will ask if treatment 1 is in the test. If yes, then it will then ask the partition value for treatment 1, whether successes lie above or below that value. The same questions will be asked for each treatment. After each treatment is either excluded or given a partition value, Lochinvar will offer the user the chance to exclude blocks from the test.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic T will be listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculation of Block and Treatment Totals

If the user chooses the method of cell counts (paragraph 2, above), Lochinvar will sum the successes over each of b blocks and over each of k treatments. These totals will be in the cells vector of line 3a of the options file (see Appendix D for details of the options file), to be read by Crunch.

If the user chooses to enter partition values (paragraph 3 above), Crunch uses that information to compute the block and treatment totals.

b. Computing the Test Statistic

The numerator of the test statistic involves the sums of successes for each treatment. The total number of successes N is found by summing each treatment number of

successes Ci; N divided by the number of treatments k is the average number of successes per treatment. Find the sum of squared deviations by summing over k treatments the square of (Ci-N/k). Multiply that sum of squares by k(k-1) to get the test statistic numerator.

The denominator of the test statistic is the sum over all b blocks of the number of successes in block i times the number of failures in block i.

Divide the numerator by the denominator.

2. Program Method of Operation

a. When Using Cell Counts

In the procedure Cochran, Lochinvar prompts the answer of yes or no to the question of whether block i, treatment j, is a success. If yes, it increments the total of successes in the count for block i and treatment j. The total for block i is contained in the ith element of the vector CELLS; the total for the jth treatment is contained in the (j+b)th element of CELLS, where b is the number of blocks. The number of blocks (NROW) and number of treatments (NCOL) are written in Line 3 of the options file; the vector CELLS is written in Line 3a. The Crunch subroutine Celler reads those values, and sends them to the subroutine Cochrn, which computes the test statistic.

b. When using Partition Values

In the procedure Cochran, Lochinvar asks the partition value of each treatment included in the test; those values are entered in the odd numbered elements of vector PART_V, that is, 1,3,...,(2k-1), where k is the number of treatments. In the even numbered elements of PART_V is a +1.0 if successes are below the preceding partition value, or -1.0 if the successes are above it. For example, PART_V of 3.0 -1.0 7.0 1.0 means that there are two treatments; in the first treatment, successes are above 3.0, in the second, successes are below 7.0. The number of

blocks (NROW) and the number of treatments (NCOL) are written in the Line 3 of the options file; PART_V is written into line 3a.

The Crunch subroutine Celler calls the subroutine Coch to use PART_V and the data elements values to find the totals of successes for the blocks and treatments. It places those counts (in the same format as CELLS, describe in subparagraph a, above) into the vector RCCT. That vector is used by the subroutine Cochrn to compute the test statistic.

3. Program Characteristics

a. Limitations

When the user enters cell counts, the sum of the treatments and blocks cannot exceed 50 (b+k), and the total number of cells (b*k) cannot exceed 400.

When the program computes the treatment and block totals, the maximum number of data points is 400; the maximum number of treatments is 15. The sum of treatments and blocks (b+k) cannot exceed 50.

b. Subroutines/Procedures Used

(1) Lochinvar: Cochran, Cel_test

(2) Crunch: Cochrn,Celler

(3) External: Shsort

XXIV. KOLMOGOROV TEST

A. OVERVIEW

1. General Purpose

With a sample of 30 or less data points, test to see whether the population from which the sample is drawn is of a specified distribution with specified parameters. The choice of hypothesised distributions is uniform, normal, exponential, weibull, and erlang (gamma with an integer shape parameter).

The Lilliefors test is identical to the Kolmogorov test except that it estimates parameters from the sample; normal and exponential distributions can be hypothesised.

2. Level of Measurement

Interval

3. Assumptions

The sample is a random sample from the population.

4. Hypotheses

With $F^*(x)$ as the theoretical distribution with specified parameters, and $S(x)$ as the population's underlying distribution function, the hypotheses are:

a. Two tailed test

(1) Null hypothesis h0: $F^*(x)=S(x)$ for all x

(2) Alternative hypothesis h1: $F^*(x)\neq S(x)$ for at least one x in range of sample

b. One tailed test

(1) Null hypothesis h0: $F^*(x)\leq S(x)$ for all x

(2) Alternative hypothesis h1: $F^*(x)>s(x)$

c. One tailed test

(1) Null hypothesis h0: $F^*(x)\geq S(x)$

(2) Alternative hypothesis h1: $F^*(x)<S(x)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If a data set has more than one variable, the user must specify which is used. If the chosen variable has more than 30 elements, the chi-square goodness of fit test will automatically be given. Otherwise, the user must specify the distribution to use, and all its parameters, from among the following:

a. Normal (specify mu, sigma squared)
b. Exponential (specify lambda=rate,scale)
c. Uniform (specify a=lower bound,b=upper bound)
d. Erlang (specify n=integer,shape; lambda=scale)
e. Weibull (specify alpha=shape, lambda=scale)

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 20. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 20. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index of variable to be used in test.
(2) Enter number of theoretical distribution.
(3) Enter parameters as prompted.
(4) Enter letter of type hypothesis.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The values of the largest difference F*(x)-S(x), the smallest (most negative) difference, and the greatest absolute difference are displayed. Depending on the hypothesis, Crunch selects the proper value of these three as the test statistic, and displays that value.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Empirical Cumulative Distribution Function (CDF)

Sort the data elements, giving order statistics X(1),...,X(n), where n is the number of data elements. The empirical cdf equals 0 for values of X lower than the smallest X, X(1). At X(1), the cdf jumps to 1/n; the cdf increases by 1/n at each new order statistic. At X(n), the empirical cdf equals 1, and remains 1 for all values of x greater than the sample. The empirical cdf is stored in the vector S.

b. Points on the Theoretical CDF

With a specified distribution and parameters, find the value of the cdf for each value of the order statistics X(1),...,X(n). For example, with exponential (lambda=2) specified and X=1.5, the value of the theoretical cdf is 1.0-exp(-(2*1.5)) =0.9502. Store these values in the vector F.

c. Difference between Empirical and Theoretical CDF's

With a sample of size n, there are 2n differences between S and F. For example, at X(1), S jumps from 0 to 1/n, so that F(1) is compared to both values. At each order statistic value, the theoretical distribution F(i) is compared to S(i-1) and S(i). Crunch keeps track of the tmin (most negative difference, or if all are positive, the smallest positive), and tmax (most positive, or if all are negative, the closest to zero).

d. Test Statistics

For a two tailed test (hypothesis A), the test statistic is the larger of the absolute values of tmax and tmin. For hypothesis B, tmax is the test statistic. For hypothesis C, tmin is the test statistic.

2. Program Method of Operation

The subroutine Kolmog compute the empirical cdf, by a do for loop i=1 to n, with S(i)=i/n. Shsort orders the sample values to form order statistics. The theoretical cdf's are found in one of the following subroutines: normf, expf, unif, erlf, and weibf.

The subroutine Findt finds the differences between S and F for each value of X. For i=1, F is compared to 0 and S(1). Then, in a do for loop i=2 to n, F(i) is compared to S(i) and S(i-1). These 2n difference are stored in the vector T. Another do for loop, j=1 to 2n, finds tmax and tmin. From those two values, the greatest absolute difference is found.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Kolmog, Upar, Wpar, Erpar, Npar, Expar

(2) Crunch: Kolmog, Findt; Normf, Unif, Expf, Erlf, Weibf

(3) External: Shsort

XXV. LILLIEFORS TEST

A. OVERVIEW

1. General Purpose

With a sample of less than 31 elements, this tests whether the sample came from a specified distribution, either a normal or exponential. The parameters used for the test are the maximum likelihood estimates (MLE), derived from the sample mean and variance. Those two values are computed and displayed in the output file; however, Lilliefors tests only the distribution type, not the parameter value. (Any value other than the MLE would have a higher probability of rejecting the null hypothesis).

If specific parameters must be tested for, use the Kolmogorov test.

2. Level of Measurement

Interval

3. Assumptions

The data values are a random sample of the population.

4. Hypothesis

With $F^*(x)$ as the hypothesised distribution function, with MLE parameters, and $S(x)$ as the actual population distribution

a. Null hypothesis

H0: $F^*(x)=S(x)$ for all x

b. Alternative hypothesis

H1: $F^*(x)\neq S(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If more than one variable is in the data set, the user must specify which one to utilize for the test. If that variable has more than 31 elements, the goodness of fit test will be offered automatically. Otherwise, choose whether to test for the normal or the exponential distribution.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 21. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 21. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

(1) Enter index of variable to be used in test, if there is more than one in the data set.

(2) Enter whether to test against the best fitting normal or exponential distribution.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The sample mean and variance are listed, as well as the test statistic.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Finding the MLE Parameters

The sample mean and variance are computed. If the test is for normality, the theoretical distribuition's parameters mu and sigma squared are estimated by the sample mean and variance respectively. If the exponential is the theoretical distribution, its parameter lambda is the reciprocal of the sample mean.

b. Calculation of the Test Statistic

The remainder of the test is exactly the same as the two- tailed Kolmogorov test. The empirical distribution function is generated using (1/n) increments for each of the n sample data elements. The difference between the empirical and theoretical distribution is computed at each value of x contained in the sample. The greatest absolute difference is the test statistic.

2. Program Method of Operation

The subroutine Lilief calls Muvar to find the sample mean and variance, then assigns MLE values to the rp vector. The subroutines Norf or Expf, as appropriate, are called to generate values of the theoretical distribution at the values of X(1),...X(n). The subroutine Findt finds the largest, smallest and greatest absolute difference; that last value is the test statistic.

3. Program Characteristics

a. Limitations

Maximum number of data points: 30, in one variable.

b. Subroutines/Procedures Used

(1) Lochinvar: Lillie

(2) Crunch: Lilief,Muvar,Expf,Norf,Findt

(3) External: Shsort

XXVI. SHAPIRO WILK TEST FOR NORMALITY

A. OVERVIEW

1. General Purpose

With a sample of 30 or less data points, test to see if the sample is from a normal distribution of unspecified parameters. Although Crunch computes the sample mean and variance, they are not displayed. No statement about the population parameters is made by this test.

2. Level of Measurement

Interval

3. Assumptions

The sample is randomly drawn from the population.

4. Hypothesis

With $F^*(x)$ as a normal distribution of unspecified parameters, and $S(x)$ as the distribution function of the population,

a. Null hypothesis

$H0: F^*(x)=S(x)$ for all x

b. Alternative hypothesis

$H1: F^*(x)\neq S(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If there is more than one variable in the test, the user must specify which one is to be utilized. If that variable has more than 30 elements, the goodness of fit test will be automatically offered. If not, no further choices are necessary.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 22. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 22. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

Enter index of variable to be used in test, if there is more than one in the data set.

4. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

5. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic will also be included.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic/Level of Significance

a. Compute the Sum of Squared Deviations

The sum of squared deviations is the sum over n of the square of the difference between X(i) and the mean of X. The program actually figures the sample variance and multiplies by (n-1). This is the denominator of the test statistic.

b. Compute the Sum of the Weighted Differences

The sample is sorted into the order statistics, X(1),..., X(n). Subtract values with the same depth from the end points,e.g, subtract X(n) - X(1), X(n-1) - X(2),...,X(n-i+1) - X(i). The n is even, there are n/2 differences; if n is odd, there are (n-1)/2 differences, with the median excluded. Multiply these differences by a(i) weights, provided by Lochinvar. Sum the weighted differences. This is the numerator of the test staistic.

c. Compute the Test Statistic

Divide the sum of weighted difference by the sum of squared deviations.

2. Program Method of Operation

The subroutine finds the sample variance by calling Muvar. Shsort is used to order the sample into order statistics X(1),...,X(n). A do for loop, i=1 to trunc(n/2), is used to calculate the sum of a(i)*(X(n-i+1)-x(i)); the a(i)'s are contained in the rp vector.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables/treatments: 15.

b. Subroutines/Procedures Used

(1) Lochinvar: Wilkes

(2) Crunch: Wilksh,Muvar

(3) External: Shsort

XXVII. SMIRNOV TEST

A. OVERVIEW

1. General Purpose

With two samples, X and Y, each drawn from a different population, test whether the underlying population distributions F(x) and G(y) could be equal over the entire range of X and Y.

The Smirnov and Cramer-von Mises test differ only in the computation of the test statistic, and therefore, the choice of hypotheses.

2. Level of Measurement

Ordinal

3. Assumptions

The samples are drawn randomly from their populations, and are rankable against each other as well as within samples.

4. Hypotheses

a. Two tailed test

(1) Null hypothesis h0: $F(x)=g(x)$ for all x

(2) Alternative hypothesis h1: $F(x)\neq g(x)$ for at least one x in range of sample

b. One tailed test

(1) Null hypothesis h0: $F(x)\leq g(x)$ for all x

(2) Alternative hypothesis h1: $F(x)>g(x)$

c. One tailed test

(1) Null hypothesis h0: $F(x)\geq g(x)$

(2) Alternative hypothesis h1: $F(x)<g(x)$

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

The user must designate which of two variables are used, if the data set contains more than two variables. The user specifies the type of hypothesis.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for a detailed explanation of the following.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name; when offered a test, type 23. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 23. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar: Enter indices of variables to be used in test.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. The test statistic will also be listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculate the Difference Vector S

X and Y are sorted into order statistics, X(1),...,X(m), and Y(1),...,Y(n). The counter for X is i, the counter for Y is j. Compare X(1) with Y(1). If X(1) < Y(1) then increment i; if X(1) > Y(1) then increment j. The difference between the empirical distribution functions is (i-1)/m minus (j-1)/n. After each comparison of X(i) and Y(j) the appropriate counter is incremented and the difference is calculated and stored in the vector S. The counter i is incremented until it reaches (m+1); j can reach (n+1). If X(i) = Y(j), increment i, calculate and store the difference, then increment j, and compute and store the difference. There will be (m+n) elements in S.

b. Calculate the Test Statistic

For a two tailed test (hypothesis A), the greatest absolute difference is the test statistic. For hypothesis B, the largest difference is the test statistic. For hypothesis C, the smallest difference is the test statistic.

2. Program Method of Operation

The subroutine Smirn calls Shsort to rank X, then rank Y. It calls the subroutine Finds to calculate the difference vector S, then chooses the test statistic based on the hypothesis specified.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

(1) Lochinvar: Smirnov

(2) Crunch: Smirn, Finds

(3) External: Shsort

XXVIII. CRAMER-VON MISES TEST

A. OVERVIEW

1. General Purpose

With two samples, X and Y, test whether the underlying population distributions $F(x)$ and $G(y)$ could be equal over the entire range of X and Y.

The Smirnov and Cramer-von Mises test differ only in the computation of the test statistic, and therefore, the choice of hypotheses.

2. Level of Measurement

Interval

3. Assumptions

The samples are randomly drawn from the respective populations, and are independent of each other.

4. Hypothesis

a. Null hypothesis

H0: $F(X)=G(X)$ for all x

b. Alternative hypothesis

H1: $F(x)=G(x)$ for at least one x

B. ACTION REQUIRED BY USER OF PROGRAM

1. User Decisions about Program Options

If there are more than two variables in the data set, specify which two are in the test.

2. Program Activation when Using Data Values

a. Initial Choices

See section III.C for initial choice details.

In CMS, type LOCHI. If data is already in a data file, enter "Y" to first question, then enter file name;

when offered a test, type 28. If data needs to be entered, enter "N" for first question, "V" for second; when data entered, choose to do a test; when offered a choice of tests, type 28. Enter "U","V",..."Z" for desired output file.

b. Variable and Parameter Choice

When prompted by Lochinvar:

Enter indices of variables to be used in test., if the data set has more than two.

3. Test Execution

To perform the test, after the above entry of data and parameters, type CRUNCH while in CMS.

4. Output Available to the User

The program CRUNCH displays the output file to which the data has been sent, e.g., "U LISTING". The names of the data set and variables used are at the top of the output file. In addition, the test statistic is listed.

C. ACTIONS PERFORMED BY THE PROGRAM CRUNCH

1. Description of Calculation of Test Statistic

a. Calculate the Difference Vector S

This step is exactly the same (using the same subroutine) as in the Smirnov test. See section XXVII for details.

b. Calculating the Test Statistic

Find the sum of the squared elements of the S vector. With m as the number of elements of X, and n as the number of elements of Y, multiply the sum of squared S(i) times (m*n)/(m+n)**2. This is the test statistic.

2. Program Method of Operation

Thesubroutine Crmvon calls Shsort to order X and order Y into order statistics. The subroutine Finds finds the vector S. In a do for loop, i=1 to (m+n), Crmvon

squares and sums S(i). That sum is then multiplied by (m*n)/(m+n)**2.

3. Program Characteristics

a. Limitations

Maximum number of data points: 400. Maximum number of variables: 2.

b. Subroutines/Procedures Used

(1) Lochinvar: Cramvon

(2) Crunch: Crmvon,Finds

(3) External: Shsort

LIST OF REFERENCES

1. Conover, C., Practical Nonparametric Statistics, John Wiley & Sons, New York, 1971

2. Stevens, S., "On the Theory of Scales of Measurement", Science, 103: 677-680, 1946

3. Zaks, Rodney, Introduction to Pascal, Sybex, Berkeley, Ca., 1981

APPENDIX A

TUTORIAL: LINKING TO AND USING THE COMPUTER PACKAGE

A. PURPOSE

This appendix contains three things: instructions on how to link to the class disk to perform nonparametric tests; instructions on how to copy the programs onto a user's A disk; and an example of how to use the program.

B. LINKING TO THE CLASS DISK

To perform the test, a student can link to user number 2266p, read only password OBRIEN, and use the programs there to perform nonparametric tests. The steps are:

1. Linking to the Class Disk

In CMS, type "CP LINK 2266P 191 192 RR". The computer will type back ENTER READ ONLY PASSWORD. Enter "OBRIEN". After receiving back the "R" message, enter ACC 192 B/A.

2. Using the Programs

An example of the use of Lochinvar and Crunch is given below. Detailed instructions for use are given in section III and the sections on specific tests.

3. Releasing the Class Disk

After you are done, type RELEASE 192.

C. COPYING THE PACKAGE

Three files are needed to run nonparametric tests: "LOCHI MODULE", "CRUNCH TEXT", and "CRUNCH EXEC". The first two are compilations of the source code, contained in "LOCHI PASCAL" and "CRUNCH FORTRAN". To copy any of these programs:

1. Link to the Class Disk

2. Copy the File: Type XEDIT <filename> <filetype>. Type SAVE = = A in the space marked ==> at the bottom of the screen, then FILE. The file will be on the A disk.

3. Release the Class Disk

D. AN EXAMPLE OF TEST USE

To fully understand this example, read or review sections III and XXI of this text. The data set used is from Conover [Ref. 1: p.319]; the test used is the Friedman test. The example shows the program's prompting in capital letters, and the user's response in lower case, with a <== to the right.

1. Entering the Data File

a. Preliminary Steps

There are four initial steps to perform before entering a data file. The first is to activiate Lochinvar, then to request not to use a previously entered file, then to enter the file by data values, and finally to write the information to data file "lochi a".

```
lochi  <== (1)
 EXECUTION BEGINS...
DO YOU WANT TO PERFORM A TEST ON A PREVIOUSLY ENTERED DATA SET?
ENTER Y OR N
   n  <== (2)

YOU HAVE A CHOICE OF TWO METHODS TO ENTER DATA:
1. ENTER EACH VALUE OF DATA SET.  VALUES MUST BE  NUMERICAL, AND
MEASUREMENT AT LEAST ORDINAL.  ALL TESTS MAY BE PERFORMED THIS WAY.
2. ENTER ONLY THE COUNT OF SUCCESSES/FAILURES OR ROW/COLUMN CELL
COUNT.  DATA VALUES AND LEVEL OF MEASUREMENT MAY BE NOMINAL, BUT
COUNTS MUST BE NON-NEGATIVE INTEGERS. THE FOLLOWING TESTS MAY BE
PERFORMED: BINOMIAL, MCNEMAR, R*C CONTINGENCY, CHI-SQUARE
GOODNESS OF FIT.
     ENTER V FOR VALUE OR C FOR COUNT
         v <== (3)

ENTER FILETYPE (A-F) TO WHICH DATA GOES
  a <== (4)
```

b. Enter Information of Label of Data Set

The label needs three types of information: the data set name, the number of variables, and the variable

names. For this example, we are using the data of exercize 1, page 309 in Conover, with four variables, named the names of the seasons:

```
 ENTER NAME OF DATA SET
  prob 1,p 309, number of births by hospital  <== (1)

 ENTER NUMBER OF VARIABLES OR TREATMENTS
    4 <== (2)

 ENTER NAME OF VARIABLE NR              1
    winter  <==   (3a)
 ENTER NAME OF VARIABLE NR              2
   spring   <==   (3b)
 ENTER NAME OF VARIABLE NR              3
   summer   <==   (3c)
 ENTER NAME OF VARIABLE NR              4
   fall     <==   (3d)
```

c. Enter Data Elements

Next, enter the data when prompted. Enter either integers or real numbers, i.e., with or without decimal points; the program will read them as reals. Leave a space (not a comma) in between numbers. Enter all elements of a variable on one line if possible; if not, there will be a chance to add more elements after the "ENTER" key is pressed. There will also be a chance to check or correct entries. In the following example, a mistake is made, then corrected, in entry of variable one. In variable two the data is checked and found correct; in variable three it is not checked. In variable four, the first four data elements are listed, the "ENTER" key is pressed, the last three data elements are added, and the data checked.

(1) Enter, Check, and Correct the Values of Variable One:

```
  WHEN ENTERING DATA ELEMENTS <1, START WITH 0.
 ENTER VALUES FOR VARIABLE              1
    92 9 98 19 21 58 43   <==
 ADD MORE ELEMS TO SAME VARIABLE?
 OR CHECK THIS VAR? OR GO TO NEXT?
 ENTER A(ADD),C(CHECK)OR G(GO ON)
   c  <==
 NUMBER OF ELEMS ENTERED              7
 NUMBER OF ELEMS IN THIS VAR              7

 92.000   9.000  98.000  19.000  21.000  58.000  43.000
```

(The last entry should have been '42' vice '43', so re-enter variable:)

```
WANT TO RE-ENTER THIS VARIABLE?
ENTER Y OR N
  y  <==
ENTER VALUES FOR VARIABLE                  1
  92 9 98 19 21 58 42
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A(ADD),C(CHECK)OR G(GO ON)
  c  <===
NUMBER OF ELEMS ENTERED                7
NUMBER OF ELEMS IN THIS VAR                7

92.000   9.000  98.000  19.000  21.000  58.000  42.000
WANT TO RE-ENTER THIS VARIABLE?
ENTER Y OR N
  n  <==
```

(2) Enter, Check Values for Second Variable:

```
ENTER VALUES FOR VARIABLE                  2
  112 11 109 26 22 71 49  <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A(ADD),C(CHECK)OR G(GO ON)
  c   <==
NUMBER OF ELEMS ENTERED               14
NUMBER OF ELEMS IN THIS VAR                7

112.000  11.000  109.000  26.000 22.000  71.000  49.000
```

(This time, it is right the first time, so no re_entry:)

```
WANT TO RE-ENTER THIS VARIABLE?
ENTER Y OR N
  n   <==
```

(3) Enter Values for Third Variable:

```
ENTER VALUES FOR VARIABLE                  3
  99 10 92 19 23 51 44  <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A(ADD),C(CHECK)OR G(GO ON)
  g<==
```

(4) Enter, Add to, Check Values for Fourth Variable:

```
ENTER VALUES FOR VARIABLE                  4
  77 12 81 18   <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A(ADD),C(CHECK)OR G(GO ON)
  a   <==
ENTER VALUES FOR VARIABLE                  4
    24 62 41    <==
ADD MORE ELEMS TO SAME VARIABLE?
OR CHECK THIS VAR? OR GO TO NEXT?
ENTER A(ADD),C(CHECK)OR G(GO ON)
  c   <==
NUMBER OF ELEMS ENTERED               28
NUMBER OF ELEMS IN THIS VAR                7

77.000  12.000  81.000  18.000  24.000  62.000  41.000
WANT TO RE-ENTER THIS VARIABLE?
```

```
 ENTER Y OR N
   n  <==
```

(5) Lochinvar offers a chance to re-enter the entire data set

```
     YOU HAVE ENTERED  4 VARIABLES,
WITH THE FOLLOWING NUMBER OF ELEMENTS IN EACH
  VAR NR          NR OF ELEMENTS
            1              7
            2              7
            3              7
            4              7
 DO YOU WISH TO RE-ENTER  THE ENTIRE DATA?
 ENTER Y OR N
   n  <==
```

(6) Lochinvar Determines the Variables are of equal size:

```
 THE VARIABLES/TREATMENTS ARE OF EQUAL SIZE:
 PAIRED CCMPARISON TESTS ARE POSSIBLE
```

(7) Choice to do a Test with this DATA

```
 NOW, DO A TEST WITH THIS DATA SET?
 ENTER Y OR N
   y  <==
```

2. Choice of Test, Test Parameters, Output File

a. Choice of Output Files

The relevant choices are 'U LISTING','V LISTING',...,'Z LISTING'. Anything that is already in the file will be written over.

```
 WHICH NAME FOR OUTPUT FILE, U-Z?
   u  <==
```

b. Choice of Test

Lochinvar will offer test according to the number of variables in the data set and whether the variables are of equal length. Test number 17 is desired; the Friedman rather than the Durbin is chosen.

```
 PICK AMONG THE FOLLOWING NON-PARAMETRIC TESTS:
1. BINOMIAL
2. QUANTILE
3. COX-STUART TEST FOR TREND <2 VARS,PAIR>
4.MANN-WHITNEY
5.SIGN <PAIR>
6.MCNEMAR <PAIR>
7.R*C CONTINGENCY TABLE (CHI-SQUARE) <PAIR>
8.GOODNESS OF FIT, N>14 (CHI-SQUARE)
9.WILCOXON SIGNED RANK <PAIR>
10.RANK CORRELATION (KENDALL & SPEARMAN) <PAIR>
11.NON-PARAMETRIC LINEAR REGRESSION <PAIR>
12.NON-PARAMETRIC MONOTONE REGRESSION <PAIR>
```

```
13.MEDIAN
14.KRUSKAL/WALLIS
15.SQUARED RANK TEST FOR EQUAL VARIANCE
16.HARTLEY TEST FOR EQUAL VARIANCE
17.FREIDMAN  OR DURBIN <BLOCKED>
18.QUADE <BLOCKED>
19.COCHRAN <BLOCKED>
20.KOLMOGOROV(ONE SAMPLE:PARAMETERS SPECIFIED
21.LILLIEFORS (ONE SAMPLE;NORMAL OR EXPONENTIAL)
22.WILKES-SHAPIRO TEST FOR NORMALITY
23.SMIRNOV (2-SAMPLE)
24.CRAMER-VON MISES (2-SAMPLE)
DO YOU WANT TO SEE THE LIST AGAIN?
 ENTER Y OR N
   n  <==
ENTER THE NUMBER OF A TEST LISTED ABOVE
   17  <==
 THIS WILL ALLOW THE FRIEDMAN OR THE DURBIN TEST
<DURBIN IS INCOMPETE BLOCK DESIGN: NEEDS FILLER>
  DO YOU WANT TO PERFORM THE FRIEDMAN TEST?
  ENTER Y OR N
     y  <==
```

c. Choice of Test Parameters

For this example, only the first, second and fourth variables are used in the test, and the third block of the data set (i.e., the third element of each variable) is excluded from the test. The final test parameter entered is the level of significance.

(1) Variables Included in Test

```
 WHEN ENTERING PARAMETERS <1, START WITH 0.
 DO YOU WANT ALL   4 INCLUDED IN TEST
 ENTER Y OR N
   n  <==
 HOW MANY VARIABLES ARE INCLUDED IN TEST?
   3  <==
 ENTER INDICES OF VARIABLES TO BE IN TEST
 ENTER INDEX OF INCLUDED VARIABLE  1
   1  <==
 ENTER INDEX OF INCLUDED VARIABLE  2
   2  <==
 ENTER INDEX OF INCLUDED VARIABLE  3
   4  <==
```

(2) Exclude Third Block

```
 DO YOU WISH TO EXCLUDE ANY BLOCKS FROM TEST
 ENTER Y OR N
   y  <==
 HOW MANY BLOCKS DO YOU WISH TO EXCLUDE?
 UP TO   7 BLOCKS
   1  <==
 IN ASCENDING ORDER, ENTER INDICES, 1 -   7
 ENTER INDEX FOR EXCLUDED BLOCK NR  1
   3  <==
```

(3) Specify Level of Significance

```
 ENTER LEVEL OF SIGNIFICANCE, ALPHA
 ENTER NUMBER>0, IF <1, START WITH 0.
   0.05
```

3. Run Test with Program Crunch

   In CMS, type CRUNCH

4. Check Output File

   Crunch will comput the test statistic,then display:

```
THE RESULTS ARE LISTED IN FILE
   U LISTING
```

To check the results, type XEDIT U LISTING

APPENDIX B

THE PROGRAM LOCHINVAR

A. Purpose

This appendix is intended to assist in modifying the program Lochinvar, or in implementing it on a computer system other than the IBM-3033 at the Naval Postgraduate School. The last part of this appendix, general description of program operation, may be of interest to a user who is experienced in the use of Lochinvar.

Section III of the text describes the general use of the program, while Sections V through XXVIII give specific instructions on responding to Lochinvar's promptings for the various tests. Appendices C and D explain in detail Lochinvar's two output files, the data file and the options file, respectively.

The source code is contained in Appendix H.

B. DESCRIPTION OF PROGRAM CHARACTERISTICS

1. Language

Lochinvar was originally written and tested in Waterloo Pascal; it is presently in a form compatible with the IBM virtual system Pascal compiler, PASCALVS. The program's syntax is consistent with the Pascal described by Zaks [Ref. 3].

2. Input and Output Files

In Lochinvar, input or output files are referred to by one of nine text definitions. "Qin" refers to terminal input; "qout" refers to terminal input; "op" refers to the options file, "lochi op". Text "a" through "f" refer to data files "lochi a" through "lochi f"; these data files are used both as input and output.

Prior to using a file for output, Lochinvar must open that file by the "rewrite" command. For the data files, this occurs in the procedure New_data; for the options file, it occurs in the procedure File_io. All seven of those files are explicitly defined as fixed format, logical record length 80. The form of the rewrite command, here illustrated for "lochi a", is:

```
REWRITE(A,'NAME = LOCHI.A.A,RECFM=F,LRECL=80')
```

The terminal output is opened in the main program, using the command "TERMOUT(QOUT)".

Prior to reading a file as input, Lochinvar must open that file for reading with the "reset" command. For the data files, this occurs in the procedure Check_file. The terminal input file must be reset prior to each time data is entered from the keyboard. Rather than having reset commands scattered throughout the program, the procedure Rst is invoked. Rst resets the terminal for input with the command "TERMIN(QIN)".

3. Dimensions of Data Structures

At the beginning of Lochinvar are six constants. Tot_data defines the total number of data elements contained in the data set; it is currently set at 400. Tot_vars is the maximum number of variables in a data set plus one; it is currently set at 16 (for 15 variables). L_names is the length of variable or data set names; it is currently set at 50. L_r_par, L_i_par, and L_e_b are the number of real parameters, the number of integer parameters, and number of excluded blocks, respectively; each is currently set at 50.

Changing the setting of any one of those numbers will change all subsequent dimensioning within the program. It is not recommended that L_names be increased beyond 80, since this would place names on more than one line (card image) of the data or options file.

4. Type and Variable Declarations

Below the constant declarations, the type declations define arrays, sets and types. Below that are variable declarations. All parameters are defined globally.

5. Description of Variables, Arrays, Sets

Below the variable declarations, a comment section gives brief descriptions of each variable, set or array passed as a parameter.

6. Description of Procedures and Functions

Each subprogram element contains a brief comment section describing its purpose.

C. DESCRIPTION OF PROGRAM OPERATION

The main program offers the user three choices: use a previously enter data set to perform a test, enter a new data set, or enter cell counts. See Section III of the text for a complete explanation of these choices. Depending on the user's choice, the main program will call one of three master subprograms to prompt the user through the data and parameter entries:

1. New_Data

The master subprogram New_Data prompts entry of a new data set and offers the user the chance to perform a test with the newly entered data. Steps in this process:

a. Data Set File Type

New_data asks the filetype for the data, "a" through "f".

b. Names and Number of Variables

New_data calls Namer, which asks the data set name, the number of variables, and the variable names.

c. Data Element Values

If there are two variables, New_data offers the choice of entering data pair by pair. If the user chooses to

do this, New_data calls E_pairs. Otherwise, the procedure E_by_var prompts entry of data variable by variable.

d. Writing of Data File

New_data writes the data set file.

e. Test Choice

New_data asks if a test should be performed with this data. If yes, File_io and Which_test are called; these procedures are described in the next paragraph.

2. Old_data

If the user indicates that a previously entered data set is to be used, the procedure Old_data is called. It performs the following:

a. Data File Choice

Old_data asks the user's choice of file type of old data, "lochi a" through "lochi f". Old_data calls Chech_file, which read and displays the name of the indicated data set. If it is the correct data set, it also reads the number of variables and number of elements per variable.

b. Output File Choice

The procedure File_io asks to which output file the test results should be sent. It writes the number of the data set file (NIF) and the number of the test results file (NOF) to line one of the options file.

c. Test Selection and Parameter Choice

Old_data calls the procedure Which_test. This procedure offers some or all of the nonparametric tests, depending on the number of variables in the data set and whether the number of elements per variable are equal. Once the choice is made, Which_test writes that choice to line 2 of the options file, along with whether the test involves cell counts. Which_test then calls the procedure for the chosen test.

d. Test Variable and Parameter Selection

Each test procedure is somewhat different, however, there are common points. If the test involves cells, the test procedure prompts cell related parameters, then calls W_cells to write those parameters to line 3 of the options file (see Appendix D for details of that file). If the data set contains more variables than the minimum needed for the test, Which_vec will prompt the user to enter which variables are included in the test. If the test involves paired or blocked data, the procedure Exclud_b offers the user the chance to exclude pairs or blocks from the test; in other cases, None_ex writes a "0" to line 5 of the options file. The test procedure then prompts entry of various integer and real parameters, and calls the procedure W_param to write those parameters to lines 6 and 7 of the options file.

3. Cel_test

If the user specified to enter cell counts, the procedure Cel_test is called by the main program. It calls File_io to write the NIF and NOF to the data file. It calls Namer to get the data set and variable names. The choice test dictates which procedure prompts the cell count entry. Cel_test writes the data set name and variable names to line 8 of the options file.

APPENDIX C
THE DATA FILE

A. PURPOSE

This appendix explains the structure of the data files, which contain the data set and variable names, the number of variables and number of elements per variable, and the data elements of the variables. This will aid the user in understanding the computer package. More importantly, it will allow the user to test data generated by another program, or read from cards or tape, without having to load the data into Lochinvar by hand.

B. METHOD OF WRITING DATA FILES

1. Formats of Data File Components

All names are written in character fields of length 48, from the first to the forty-eighth position on a line. All integers are written in a 16I5 format, i.e., with the least significant digit of the integer on position 5,10,...,80. All data elements are read as real numbers; they are written to be read in an 8F10.3 format. This means that there are eight fields for real numbers per line, 1-10,11-20,...,71-80, and the real numbers have three digits to the right of the decimal point, e.g., 789.123. Lochinvar includes a decimal point in all real numbers; if this is not possible, the least significant digit of an entry (the 0.001 digit) must be right justified in the field, i.e., aligned on column 10,20,...,80.

If the user has a data set whose data elements are all integers, there is a way to have the data file written without converting all the integers to real numbers. The least significant digit of the integers can be aligned on

columns 7,17,...,77, and the program Crunch will read them as real numbers with the same value as original integer data.

If the data elements of the data set are either all very large or all very small, they may not fit into the F10.3 format. A quick fix to that problem is to rescale the values, e.g., multiply very small numbers by 1000, or divide large numbers by that amount. The validity of the statistics will not be effected by that transformation.

If there are both very large and very small numbers in the data set, there will be numerical errors in single precision Fortran for any tests which use an interval level of measurement. For tests which use ordinal or nominal level of measurement, any monotonic transformation, such as y=ln x, could be used to allow both very large and very small data values to fit into the F10.3 format.

2. Layout of the Data File

a. Data Set Name

The data set name occupies the first line of the data file. The name is from 1 to 48 characters long. On the same line as the name, in the forty-ninth position, there is the character "|", which on IBM keyboards is the upper case "1". For example, a data set file name:

```
This is a data set name                         |
```

b. Number of Variables and Elements per Variable

On the line following the data set name, there is a line of integers. The first integer is the number of variables in the data set. The second integer is the number of elements in the first variable; the third integer is the number of elements in the second variable, and so on. There must be non-zero integers in the first two positions. Lochinvar writes zero where there are no values, but that is not necessary. As mentioned above, integers are right justified on position 5,10,...,80.

Let the data set consist of two variables. The first variable has two data elements, the second has four. the first two lines would appear as:

```
This is a data set name                       |
 2    2    4    0    0    0    0    0...
```

c. Data elements

The data elements are listed variable by variable, i.e., all the elements of the first variable are listed, then all of the second, et cetera. As mentioned above, the data elements are read as real numbers in the F10.3 format.

Let the elements of the first variable be 27.5 and 31; let the elements of the second variable be 13, 17, 25 and 35. The first three lines of the data file would appear as:

```
This is a data set name                       |
 2    2    4    0    0    0    0    0    0    0    0...
 27.500    31.000    13.000    17.000    25.000    35.000
```

The zeros to the right of the decimal point are written by Lochinvar but are not necessary. The number could have been written without decimal points, with proper alignment, as shown below. However, when numbers are written with decimal points, a mistake in alignment is no problem; without decimal points, alignment is crucial. Using only alignment for data elements:

```
This is a data set name                       |
 2    2    4    0    0    0    0    0    0    0    0...
  275       31        13        17        25         35
```

d. Variable names

The variable names begin on the line directly following the last data element line. Each variable name occupies a line; there must be the same number of lines for variable names as there are number of variables indicated in line two.

Let freshman be the name of the first variable and senior be the name of the second.

```
   This is a data set name
    2    2    4    0    0    0    0    0    0    0    0...
    27.500    31.000    13.000    17.000    25.000    35.000
  freshman
  senior
```

C. LOCATION ON THE USER'S A DISK

1. Default Location

As mentioned repeatedly throughout the text, by default the data files are named "lochi a",..."lochi f". Lochinvar writes files to those six locations; Lochinvar will not be easily modified unless the user is experience in programming Pascal.

2. Reading Files from other than Default Locations

The program Crunch reads "lochi a",...,"lochi f" based on the file definitions in the file "CRUNCH EXEC" and the number of input file (NIF) of the options file. "CRUNCH EXEC" is explained in Appendix E, while the input file is explained in Appendix D. To have Crunch read from a properly written data file named anything other than "lochi a",...,"lochi f", the user will have to do the following:

a. Write an options file for the data (see section III and Appendix D), either manually (not recommended), or else using Lochinvar, specifying a data set with the same general characteristics as the test data set. For example, if the test data set is named QUEUE DATA, with four variables of equal length, tell Lochinvar to run the desired test on data set "lochi a", given that "lochi a" has four variables of equal lenghth

b. Since the number of input file written in the options file is "1", corresponding to the default location "lochi a", change the file definition in the file "CRUNCH EXEC". The origin file "CRUNCH EXEC",

```
&TRACE
&ERROR &EXIT 99
FILEDEF 01 DISK LOCHI A
...
LOAD CRUNCH (START
```

becomes,

```
&TRACE
&ERROR &EXIT 99
FILEDEF 01 DISK QUEUE DATA
...
LOAD CRUNCH (START
```

c. Alternate Method

It would be easier, in the example above, to rename the current file "lochi a" to some other name, and rename "QUEUE DATA" to "lochi a".

APPENDIX D

THE OPTIONS FILE

A. PURPOSE

This appendix describes the options file. The twenty nonparametric tests which require ordinal or interval data level of measurement all have options files with similar format. The four tests which can handle nominal data have slightly idiocyncratic formats, which contain all the information of other test plus something additional.

This appendix is included for completeness of documentation of this text. It is not recommended that a user by-pass Lochinvar to write the options file by hand. If the computer package is implemented on an operating system which cannot run Pascal, though, that may be necessary.

B. GENERAL INFORMATICN ABOUT THE OPTIONS FILE

1. Name

Lochinvar writes the options file to "lochi op" on the user's A disk.

2. General Format Rules

As in the data file, all integers are written in the format 16I5, all real numbers are written as 8F10.3, all names are in 48 character lines. See Appendix C, paragraph B. , for all the different ways to have real numbers represented for the program Crunch to read.

3. Explanation of the Options File Format

a. List of Line Numbers

There are 14 line numbers: 1, 2, 3, 3a, 4, 4a, 5, 5a, 6, 6a, 7, 7a, 8, 8a.

b. Line Numbers in the Option File

Each of the 14 line numbers occupies at least one line (or card image) in the options file, unless that line is omitted. There is not a blank card for omitted line numbers; rather, the previous line is followed by the next. For example, if line numbers 3 and 3a are ommited, line 4 immediately follows line 2.

The lines with "a" in their number may occupy more than one line (or card image) in an actual file, because they may contain more information than would fit on a single 80 column card.

c. Rules for Inclusion of Lines in Options File

Lines 1 and 2 are included in every options file.

Lines 3 and 3a are omitted unless the test is one of the four which handle nominal data.

Lines 4 and 4a will be included in every test except when the user has entered cell counts (in one of the four tests which use nominal data). Line 4 is a single positive integer value which tells how many values to read in line 4a.

Line 5 equals "0" and line 5a is omitted unless the user has excluded blocks of the data set from being tested. In that case, line 5 indicates the number of elements to be read in line 5a.

Lines 6 and 7 are included in every test, and will contain single integer values. If a value of "0" is in one of those lines, then the corresponding "a" line will be omitted. For example, if line 6 has a "0", then line 6a will be omitted. If the single value in lines 4, 5, 6, and 7 is not "0", it must be a positive integer, representing the number of elements to be read in the corresponding "a" line. For example, if the single value in line 5 is "3", there will be three elements to read in line 5a.

The "a" lines may occupy as many actual lines in the data file to contain all the values to be read.

Lines 8 and 8a are used only when the user has entered cell counts for a test involving nominal data.

d. Content of the Lines

(1) Line 1: the number of input file (NIF), and number of output file (NOF), both integers

(2) Line 2: the number of the requested test (NTEST), and whether the test involves cell counts (CELLS), both integers, CELLS=1 if the test is one of the four which can use nominal data.

(3) Line 3: the cell counter option (TYPCEL), the number of rows in the cell matrix (NROW), the number of columns in that matrix (NCOL), and the number of elements in line 3a (NTORD), all positive integers. Depending on the TYPCEL option, line 3a will contain either integers or reals.

(4) Line 4: the number of treatments to be included in this test (NTREAT), a positive integer. Line 4a contains the indices of the variables of the data file which are included in the data set to be tested.

(5) Line 5: the number of blocks excluded from the test (NEXB). Line 5a contains the indices of excluded blocks.

(6) Line 6: the number of integer parameters, a non-negative integer. If line 6 is greater than zero, line 6a contains the integer parameters for the test.

(7) Line 7: the number of real parameters, a non-negative integer. If line 7 is greater than zero, line 7a contains the real parameters for the test.

(8) Line 8: data set name. Line 8a contains the names of data set variables.

C. EXAMPLE OF OPTIONS FILE FOR ORDINAL OR INTERVAL TEST

Table 5 contains a representative data file, with the line numbers of the options file for the twenty tests which require ordinal or interval data. This example is for the test to perform the problem contained in Appendix A, i.e., problem 1, page 309 of Conover [ref 1]. In this example, three of four variables were used, and one block excluded. Line number are enclosed by < >. Comments are enclosed by (*-*); the comments refer to the names of variables, as mentioned in the previous section.

TABLE 5

Options

```
<1>     1     11    (*NIF=1,NOF=11*)
<2>    17      0    (*NTEST=17,CELLS=0*)
<4>     3           (*NTREAT=3*)
<4a>    1      2     4    (*indices of treatments*)
<5>     1           (*NEXB*)
<5a>    3           (*index of the block excluded*)
<6>     0           (*number of integer parameters=0*)
<7>     1           (*number of real parameters=1*)
<7a>    0.95        (*real parameter=0.95*)
```

APPENDIX E

THE PROGRAM CRUNCH

A. PURPOSE

This appendix is intended to assist implementing Crunch on systems other than the IBM-3033 at the Naval Postgraduate School. It will also assist in modifying the program, or adapting portions of it for use in other programs.

B. DESCRIPTION OF THE PROGRAM STRUCTURE

1. General

While Lochinvar is designed to be used only as an integral unit, Crunch is separable. Therefore, many things which are centralized in the main program of Lochinvar--such as dimensioning and variable declaration--are done in each subroutine of Crunch. Explanations and documentation are also at the subprogram level.

2. Subprogram Groups

Crunch can be thought of as a system of five groups, each made up of several subroutines and functions.

a. Main Program

The main program, together with the subroutines Fname and Vname, form one group; they read the data set and variable names, call the correct data formation group, then call the correct test subroutine, and remind the user which file contains the output.

b. Data Formation Group for Cell Counts

If the test involves cell count (Binomial, Quantile, McNemar, Test for Independence, Cochran), the data formation group Celler is called by the main program. Celler and the subroutines it calls produce the cell counts used as

the raw material for calculations in the test subroutine. Those cell counts are contained in the vector RCCT, which is dimensioned by the number of rows NROW and the number of columns NCOL.

c. Data Formation Group for Data Values

For all other tests, the master reader group is called. Mstrdr and its subroutines read the specified data file and the options file. They produce a list of the data elements which are relevant for the test, i.e., Mstrdr includes only the data set variables indicated, and excludes any blocks or pairs specified. These data values are contained in the vector LIST. LIST is dimensioned by the pointer vector LPTR. LPTR(1) is always 1. LPTR(i) is the index of the first element of the ith variable in LIST. With k variables, LPTR(k+1) is one more than the last element of LIST. For example, if LIST contains three variables, with two, four and seven elements each, LPTR will have four elements, (1,3,7,14). In dereferencing variable values, the ith variable will have elements LIST(LPTR(i)) through LIST(LPTR(i+1)-1). NTREAT is the number of variables included in the test.

d. Utility Subroutines and Functions

Many subroutines are called by more than one test subroutine. These versatile subroutines are grouped together.

e. Test Subroutine Groups

Each of the 24 tests has a principle subroutine called by the main program. In general, these subroutines perform the bulk of the computations for the test statistic or level of significance. However, some of the computation is done by calling other subroutines. Subroutines which are called only by one test are listed contiguous to that test's subroutine. Subroutines which are called by more than one test are listed in the utility subroutine group.

The test subroutines make use of either LIST or RCCT in their computations.

3. Language

Crunch is written to be compatible with the IBM Fortran H compiler. Only the most rudimentary syntax was used. The only loop statements are the do for loops of the form,

```
DO 10 I=LL,UL
```

where 10 is a line label for a continue statement, LL is the lower limit and UL is the upper limit of the loop. All other loops or branching is done with if statements. This very primitive, unstructured Fortran should be compatible with virtually any computer system.

4. Input/Output Files

The IBM file definitions are contained in the file "CRUNCH EXEC". All files except 08 (terminal) are implicitly fixed format, logical record length 80. Files 01 through 05 refer to "lochi a" through "lochi e"; file 06 is "problem data"; file 07 is "lochi f"; file 10 is "lochi op"; files 11 through 16 are "u listing" through "z listing".

The number of input file (NIF) and number of output file (NOF) are listed in line one of the options file, and read by Crunch's main program. Files 01,...,05,07, and 10 are candidates for NIF; files 11,...,16 are candidates for NOF. File 08 is used only to remind the user of which output file contains the test result. File 06 contains system error messages, while file 09 contains intermediate computations of Crunch.

The main program reads lines one and two of the options file. Either Celler or Mstrdr read most of the rest of the data and options file. Fname reads the data set name and writes it to the output file; Vname does the same for the variable names.

The test subroutines write all of the test results files "u listing" through "z listing", except the data set and variable names.

5. Dimensions and Declarations

At the beginning of the main program, and of each subroutine, the variables used are declared and the arrays dimensioned. All declarations and dimensioning is explicit. Although arrays could be dimensioned by variables in subroutines, this is not done. For example, with TOTELM as the integer representing the total data elements of the test, LIST could be dimensioned in a test subroutine as LIST(TOTELM) vice LIST(400). This was not done; as explained above, Crunch is intended to be separable. To change the capacity of the program, a global change command may be issued, if the installation has a good text editor. Otherwise, the dimensions must be changed one by one.

There are a very few arrays which consume inordinate amounts of storage for what information they provide. They have the dimension of 20,000; that dimension allows storage of a comparison of each of 400 data elements with each other data element. On the IBM-3033, these large arrays require the user to request 1 megabyte of storage. On smaller systems, these arrays may be redimensioned down to exactly the size necessary (if k=the number of elements, k(k-1)/2). An alternative is to eliminate them altogether, and take out test procedures which use them.

6. Description of Variables

At the beginning of the program, there is a brief description of all variables and arrays which are used or passed by the main program. After the subprogram title line, a similar list defines the variables used exclusively among subroutines of that subprogram.

Above the variable declarations in each test subroutine, there is a description of each of the integer and real parameters used in that test.

C. EXTERNAL SUPPORT

The non-IMSL subroutine Shsort is essential to the operation of the program Crunch. Its documentation is available to any user at the Naval Postgraduate School. Any system sort routine can be used to support Crunch, which is currently configured to pass a real vector of values to be sorted, an integer vector (1,...,n) to receive the order of the original data values, and the number of elements n.

The non-IMSL Utest was used for the Mann-Whitney test. Its value is that it not only computes the test statistic, but also the level of significance. Modifications to Crunch could easily allow computation of the test statistic.

APPENDIX F

FORTRAN SOURCE CODE: CRUNCH

```
C
C THIS PROGRAM TAKES DATA FROM TEST A-F FILE, INFORMATION
C   FROM TEST OP FILE TO PERFORM NON-PARAMETRIC TESTS
C
C SUBPROGRAM GROUPS
C
C MAIN      CONTAINS THE MAIN PROGRAM, FNAME, VNAME, REMIND
C              READS TOP 2 LINES OF OPTIONS FILE, WRITES THE
C              TOP OF OUTPUT FILE, CALLS CELLER OR MSTRDR,
C              TEST SUBROUTINE; REMINDS WHICH IS OUTPUT FILE
C CELLER    CONTAINS CELLER,COCH,TC3,TC3LN,DICOT,TC4,TC4LN,
C              PAIRST,MCOUNT. FINDS THE CELL COUNTS FOR FIVE
C              NP TESTS: BINOMIAL,QUANTILE,MCNEMAR,COCHARAN,
C              AND RC CONTINGENCY.  CELL COUNTS TO RCCT;
C              DIMENSIONED BY NROW AND NCOL
C MSTRDR    CONTAINS MSTRDR,SUBLIST,NULIST,     .
C              READS THE DATA FILE NUMBER OF VARIABLES, #
C              ELEMENTS/VARIABLE, DATA ELEMENTS.  INCLUDES
C              VAR'S OF TEST INTO LIST, EXCLUDED PAIRS OR
C              BLOCKS AS REQ.  YIELDS LIST(DATA ELMS),
C              LPTR(POINTER FOR LIST), NTREAT(# OF VAR IN
C              TEST), TOTELM (# ELMS IN LIST)
C UTILITY   CONTAINS GENERAL SUPPORT SUBROUTINES/FUNCTIONS
C             SUBROUTINES: EXPF,FINDS,FINDT,MUVAR,NORMF,NORMV,
C              PARCOM,RANQUE,RHOER,RNQ,RPARAM
C TESTS     CONTAINS THE 24 NONPARAMETRIC TESTS: USE EITHER
C              EITHER LIST OR RCCT; COMPUTE TEST STATISTICS
C              OR LEVEL OF SIGNIFICANCE; WRITES TO OUTPUT
C              FILE
C              THE ORDER IN THE PROGRAM IS THE SAME AS THE
C                NUMBER OF THE TEST, NTEST
C       VARS
C       NIF       NUMBER 1-6 INDICATING FILE TEST A-F FOR DATA
C       IHOLD     INT VECTOR(16) GENERAL PURPOSE HOLDER
C       NVARS     INT NUMBER OF VARS OR TREATMENT IN DATA SET
C       NTREAT    INT NUMBER OF VAR/TREATS TEST ACTUALLY USES
C       TREAT     INT VECTOR(16) INDICES OF TREATS OR VARS
C                 ACTUALLY USED
C       NELMS     INT VECTOR(16) # ELEMENTS OF DATA IN EACH
C                         VAR/TREAT; IF SOME BLOCKS EXCLUDED,
C                         NUMBER REDUCED
C       TOTELM    INT TOTAL # OF DATA ELEMENTS IN DATA SET;
C                         IF SOME BLOCKS EXCLUDED, NUMBER REDUCED
C       NTEST     INT INDEX OF NON-PARAMETRIC TEST TO BE USED
C       EXB       INT VECTOR(200) INDICES OF BLOCKS TO BE
C                         EXCLUDED;MAY BE EMPTY
C       NEXB      INT NUMBER OF ELEMENTS OF EXB
C       IP        INT VECTOR(50) INTEGER PARAMETERS
C       IP1       INTEGER VAR FOR CELLER: MAYBE NROW OR SOME
C                         OTHER VALUE DEPENDING ON TYPE OF CELL
C                         DEFINITION (SEE TYPCEL)
C       NIP       INT NUMBER OF ELEMENTS IN IP
C       RP        REAL VECTOR(10) REAL PARAMETERS
C       NRP       INT NUMBER OF ELEMENTS IN RP
C       NIF       INPUT FILE INDEX
C       NOF       OUTPUT FILE INDEX
C       LIST      REAL VECTOR(400) DATA ELEMENTS
C       LPTR      INT VECTOR(17) POINTS TO THE FIRST ELEMENT
C                      OF EACH VAR OR  TREATMENT'S DATA IN LIST;
C                    THE LAST ELEMENT OF LPTR=TOTELM+1
C       RNVARS    REDUCED NUMBER OF VARIABLES
C       RNELMS    VECTOR OF REDUCED # OF ELEMENTS PER VARIABLE
C       DICOP     =-1 IF DICOTOMIZE BY PARTITION, ELSE=NUMBER
```

```
C                       OF VALUES OF SMALLER SUBSET OF DATA,
C                       C IF DATA NOT DICOTOMIZED
C       DIVAL    REAL PARTITION VALUE OR LIST OF SUBSET VALUES
C       NSUCC    IF DICOTOMIZED DATA, HOW MANY OF ONE SUBSET
C       CELLS    INT =1 IF TEST USES CELL COUNTS, ELSE =0
C       TYPCEL   INTEGER METHOD OF DEFINING CELLS:
C                       1-DICOT PARTITION,
C                       2-DICOT LIST,3-R*C PARTITION,
C                       4-FIXED ROW/COL
C                       COUNT, 5-ENUMERATED CELL COUNT
C       RCCT     ROW,COL CELL COUNT, BY ROWS (RE FIRST VAR),
C            E.G.,ALL COLS OF ROW 1, THEN ALL COLS OF ROW 2
C       ROWNR    VECTOR OF ROWS TO WHICH 1ST VAR ELEMS BELONG
C       COLNR    VECTOR OF COLS TO WHICH 2D VAR ELEMS BELONG
C       RORC     INTEGER LOGICAL 1=ROW 0=COL
C       NTORD    INTEGER # OF VALUES TO READ (W/I CELLER)
C       CIV      INTEGER VECTOR CELL INPUT VALUES
C       CRP      REAL VECTOR CELL INPUT VALUES
C       TIEV     INTEGER VECTOR OF TIES:TIEV(1).LE,TIEV(2).GT;
C                     BOTH  WITHIN EPSILN OF PARTITION VALUE
C-------------------------------------------------------------------
        INTEGER NIF,NFILE,IHOLD(16),NVARS,TOTELM,NTEST,
      *NTREAT,IP(50),NRP,LPTR(17),NELMS(16),RNELMS(16),
      *DICOP,NSUCC,CELLS,NOF,CIV(50),NTCRD,RCRC,ROWNR(50),
      *RCCT(50),TYPCEL,TIEV(2),EXB(200),TREAT(16),RNVARS,
      *REDELM,COLNR(50)
        REAL RP(50),LIST(400),DIVAL(200),CRP(50)
 101    FORMAT(16I5)
 102    FORMAT(8F10.3)
 201    FORMAT(1X,16I5)
 202    FORMAT(1X,8F10.3)
        READ(10,101) NIF,NOF
        READ(10,101) NTEST,CELLS
        WRITE(9,201) NIF,NOF,NTEST
        TYPCEL=0
        IF (CELLS.EQ.1) CALL CELLER(NIF,NOF,NTEST,NROW,NCOL,
      *    RCCT,LIST,LPTR,NVARS,NTREAT,NELMS,RP,IP,
      *    TREAT,TIEV,TYPCEL)
        IF (CELLS.EQ.0) CALL MSTRDR(NVARS,NELMS,LIST,
      *LPTR,TOTELM,NTREAT,NIP,IP,NRP,RP,NIF,NOF,TREAT)
        IF((TYPCEL.NE.62).AND.(TYPCEL.NE.72))
      *    CALL VNAME(NIF,NOF,NVARS,NTREAT,TREAT)
        IF(NTEST.EQ.1) CALL BINTST
      *              (NOF,RCCT,RP,IP,ALPHAT,CILL,CIUL)
        IF(NTEST.EQ.2) CALL QUANTL(NOF,RCCT,RP,IP,ALPHAT,
      *  CILL,CIUL,LIST,LPTR,TIEV,TYPCEL)
        IF(NTEST.EQ.3) CALL COXSTU(LIST,LPTR,IP,RP,NOF,NTEST)
        IF(NTEST.EQ.4) CALL MANWHT(LPTR,LIST,NOF,RP)
        IF(NTEST.EQ.5) CALL SGNT(LIST,LPTR,NOF,TOTELM,RP,IP,
      *  NTEST)
        IF(NTEST.EQ.6) CALL MCNEMR(RCCT,RP,IP,NOF)
        IF(NTEST.EQ.7) CALL RCONT(RCCT,NROW,NCOL,NOF)
        IF(NTEST.EQ.8) CALL GDFIT(LIST,LPTR,IP,RP,NOF)
        IF(NTEST.EQ.9) CALL WILCOX(LIST,LPTR,IP,RP,NOF)
        IF(NTEST.EQ.10) CALL CORREL
      *               (LIST,LPTR,NTREAT,TREAT,NOF,IP,RP)
        IF(NTEST.EQ.11)CALL REGRET(LIST,LPTR,IP,RP,TREAT,NOF)
        IF(NTEST.EQ.12)CALL MCNOFE(LIST,LPTR,IP,RP,TREAT,NOF)
        IF(NTEST.EQ.13)CALL MEDIA(NTREAT,LIST,LPTR,NIP,IP,NOF)
        IF(NTEST.EQ.14)CALL KRUWAL
      *                  (LIST,LPTR,IP,RP,NOF,NTREAT,TREAT)
        IF(NTEST.EQ.15)CALL SQUARK
      *                  (LIST,LPTR,NTREAT,TREAT,IP,RP,NOF)
        IF(NTEST.EQ.16)CALL HARTLE
      *                  (LIST,LPTR,NTREAT,TREAT,IP,RP,NOF)
        IF(NTEST.EQ.17)CALL FRIEDM
      *                  (NTREAT,LPTR,LIST,IP,RP,NOF,TOTELM)
```

```
      IF(NTEST.EQ.18)CALL QUADE
     *                   (NTREAT,LPTR,LIST,NOF,TOTELM)
      IF(NTEST.EQ.19)CALL COCHRN(NROW,NCOL,ROCT,IP,RP,NOF)
      IF(NTEST.EQ.20)CALL KOLMOG(LIST,LPTR,TREAT,IP,RP,NOF)
      IF(NTEST.EQ.21)CALL LILIEF(LIST,LPTR,TREAT,IP,RP,NOF)
      IF(NTEST.EQ.22)CALL WILKSH(LIST,LPTR,TREAT,IP,RP,NOF)
      IF(NTEST.EQ.23)CALL SMIRN2(LIST,LPTR,TREAT,IP,RP,NOF)
      IF(NTEST.EQ.24)CALL CRVONM(LIST,LPTR,TREAT,IP,RP,NOF)
      CALL REMIND(NOF)
      STOP
      END
C-----------------------------------------------------------------
      SUBROUTINE REMIND(NOF)
      INTEGER NOF
      WRITE(8,101)
 101  FORMAT(1X,' THE RESULTS ARE LISTED IN FILE')
      IF (NOF.EQ.11) WRITE(8,11)
 11   FORMAT(1X,'    U LISTING')
      IF (NOF.EQ.12) WRITE(8,12)
 12   FORMAT(1X,'    V LISTING')
      IF (NOF.EQ.13) WRITE(8,13)
 13   FORMAT(1X,'    W LISTING')
      IF (NOF.EQ.14) WRITE(8,14)
 14   FORMAT(1X,'    X LISTING')
      IF (NOF.EQ.15) WRITE(8,15)
 15   FORMAT(1X,'    Y LISTING')
      IF (NOF.EQ.16) WRITE(8,16)
 16   FORMAT(1X,'    Z LISTING')
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE VNAME(NIF,NOF,NVARS,NTREAT,TREAT)
C  THIS READS EACH VARIABLE NAME, AND IF THE VARIABLE
C IS IN THE TEST, WRITE THE NAME INTO THE OUTPUT FILE
      INTEGER NIF,NOF,NTREAT,TREAT(16),NVARS,I,J
      J=1
      IF (NVARS.EQ.NTREAT) WRITE(NOF,402) NVARS
 402  FORMAT(1X,'ALL',I5,
     * ' VARIABLES OF DATA SET USED IN THIS TEST')
      IF (NVARS.NE.NTREAT) WRITE(NOF,403) NVARS,NTREAT
 403  FORMAT(1X,'OF',I5,' VARIABLES IN DATA SET,',/,5X,I5,
     *        ' USED IN THIS TEST')
      DO 10 I=1,NVARS
       READ (NIF,200)
       IF (I.NE.TREAT(J)) GO TO 10
        WRITE(NOF,401) TREAT(J)
 401    FORMAT(1X,' THE NAME OF VARIABLE/TREATMENT #',I4)
        J=J+1
        WRITE(NOF,200)
 10   CONTINUE
 200  FORMAT (1X,
     *50HAAAABBBBCCCCDDDDEEEEFFFFGGGG0000IIIIJJJJKKKKLLLLMM)
 99   CONTINUE
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE FNAME (RORW,IF,OF)
C   THIS READS THE FILE NAME FROM FILE "IF", IF RORW IS 1,
C AND WRITES THE FILENAME TO OF IF RORW IS 2
      INTEGER RORW, IF,OF
      IF (RORW.GT.1) GO TO 10
         READ(IF,200)
         GO TO 99
 10   WRITE (OF,122)
 122  FORMAT (10X, ' NAME OF DATA SET: ')
      WRITE (OF,200)
 200  FORMAT (1X,
```

```
     * 48HAAAABBBBCCCCDDDDEEEEFFFFGGGGOOOOIIIIJJJJKKKKLLLL)
 99   CONTINUE
      RETURN
      END
C***********************************************************
      SUBROUTINE CELLER(NIF,NOF,NTEST,NROW,NCOL,RCCT,
     *LIST,LPTR,NVARS,NTREAT,NELMS,RP,IP,TREAT,TIEV,TYPCEL)
C  THIS SUBPROGRAM GROUP FINDS THE CELL COUNTS FOR ONE OF
C FIVE NONPARAMETRIC TEST--BINOMIAL, QUANTILE, COCHARAN,
C MCNEMAR, RC CONTINGECY (CHISQAURE TEST FOR INDEP)-- AND
C PLACES THOSE CELL COUNTS INTO THE VECTOR RCCT, WHICH IS
C DIMENSIONED BY NROW AND  NCOL.  UNLESS THE USER HAS
C ENTERED THE CELL COUNTS, THE SUBROUTINE CALLS SUBROUTINE
C MSTRDR TO READ THE DATA SET INDICATED BY NIF.  THE DATA
C VALUES ARE IN LIST, WHICH IS DIMENSIONED BY LPTR.
C        SUBROUTINES:
C DICOT    PERFORMS THE DICOTOMIZATION FOR THE BINOMIAL AND
C             QUANTILE, GETTING # SUCCESS=RCCT(1) AND # FAIL
C             =RCCT(2) TYPCEL=2 IF BY LIST, =1 IF BY PARTVAL
C COCH     GETS THE BLOCK AND TREATMENT SUMS OF SUCCESSES
C             BY PARTVAL; BLOCK SUCCESSES INTO RCCT(1),....,
C             RCCT(B); TREAT SUCCESS INTO RCCT(B+1),....,
C             RCCT(B+T);  TYPCEL=
C MCOUNT   GETS THE FOUR CELL COUNTS FOR THE MCNEMAR TEST.
C             RCCT(1) IS SUCCESS/SUCCESS, RCCT(2) SUCCESS/
C             FAILURE, ETC.  TYPCEL=
C TC3      DIVIDES PAIRS INTO CELLS OF AN RC CONTINGENCY
C             TABLE, BASED ON PARTITION VALUES.  RCCT(1)
C             IS FIRST ROW FIRST COL, RCCT(2) IS FIRST ROW
C             SECOND COL, ETC.  TYPCEL=3. CALLS PAIRST,
C             TC3LN, AND RCCTR
C PAIRST   SORTS THE VALUES OF THE TWO VARIABLES, AND KEEPS
C             AN INDEX KEY ON EACH, IN ORDER TO RE-PAIR
C             THE PAIRS; KEY1=ROW, KEY2=COL
C TC3LN    CALLED BY TC3.  ASSIGNS AN INTEGER CORRESPONDING
C             TO THE ROW IN WHICH AN ELEMENT OF THE FIRST
C             VARIABLE FALLS, OR THE COL IN WHICH AN ELEMENT
C             OF SECOND VARIABLE FALLS, BASED ON PARTITION
C             VALUES CONTAINED IN VECTOR CRV. INTEGER VECTOR
C             RORCNR CONTAINS THE INDICES OF ROW OR COL
C             ASSIGNED FOR EACH VALUE
C RCCTR    TAKES THE RORCNR VECTORS FOUND BY TC3LN OR TC4LN
C             AND FORMS CELL COUNTS
C TC4      DIVIDE THE DATA PAIRS INTO THE CELLS OF AN RC
C             CONTINGENCY TABLE BASED ON FIXED ROW AND COL
C             TOTALS.  TYPCEL=4. CALLS PAIRST, TC4LN, AND
C             RCCTR
C TC4LN    ASSIGNS A VALUE TO A ROW OR COLUMN, BASED ON
C             FIXED ROW OR COL TOTALS; THE VECTOR OF ASSIGN
C             IS RORCNR
      INTEGER NIF,NFILE,IHOLD(16),NVARS,TOTELM,NTEST,
     *NTREAT,IP(50),NRP,LPTR(17),NELMS(16),RNELMS(16),
     *DICOP,NSUCC,CELLS,NOF,CIV(50),NTORD,RORC,ROWNR(50),
     *RCCT(50),TYPCEL,KEY1,KEY2,K,KK,I,J,JJ,IP1,NROW,NCOL,
     *EXB(200),TREAT(16),RNVARS,REDELM,COLNR(50),T,LL,UL,
     *TIEV(2)
      REAL RP(50),LIST(400),DIVAL(200),CRV(50),VAL1(200),
     *VAL2(200)
      READ(10,101) TYPCEL,IP1,NCOL,NTORD
 101  FORMAT(16I5)
 102  FORMAT(8F10.3)
 201  FORMAT(1X,16I5)
 202  FORMAT(1X,8F10.3)
      NROW=1
      IF (TYPCEL.GT.2) NROW=IP1
      WRITE (9,209) TYPCEL,NROW,NCOL,NTORD
 209  FORMAT(1X,'TYPECEL,NROW,NCOL,NTORD,',4I5)
```

```
      T=TYPCEL
      IF ((T.NE.5).AND.(T.NE.61).AND.(T.NE.62).AND.
     *(T.NE.71).AND.(T.NE.72)) GO TO 10
         READ(10,101) (RCCT(I),I=1,NTORD)
         IF ((T.EQ.62).OR.(T.EQ.72.))
     *   CALL FNAME(1,NIF,NCF)
         LIST(1)=0.0
         LPTR(1)=1
         NVARS=IP1
         NTREAT=IP1
         IF(T.EQ.71) NTREAT=NCOL
         IF(T.EQ.71) NVARS=NCOL
         DO 5 KK=1,NTREAT
            TREAT(KK)=KK
 5       CONTINUE
         NELMS(1)=0
C        NTREAT=NCOL
C     READ(10,101) NTRETE
C     WRITE (9,401) NTRETE
C401  FORMAT(1X,'NUMBER OF TREATMENTS',I5,' INDICES')
C     READ(10,101) (TREAT(I),I=1,NTRETE)
C     WRITE(9,201) (TREAT(I),I=1,NTRETE)
C     READ(10,101) NEXB
C     WRITE (9,302) NEXB
C302  FORMAT(1X,'NUMBER OF EX BLOCKS', I5,' INDICES')
C     IF (NEXB.GE.1) READ(10,101) (EXB(I),I=1,NEXB)
C     IF (NEXB.GE.1) WRITE(9,201) (EXB(I),I=1,NEXB)
         CALL RPARAM(NIP,NRP,IP,RP)
         IF ((T.NE.62).AND.(T.NE.72.))
     *      CALL FNAME(1,10,NOF)
         CALL FNAME(2,9,9)
         CALL FNAME(2,10,NOF)
         GO TO 99
 10   IF ((TYPCEL.GT.3).AND.(TYPCEL.LT.6)) GO TO 12
         READ(10,102) (CRV(I),I=1,NTORD)
         WRITE(9,202) (CRV(I),I=1,NTORD)
         GO TO 14
 12   READ(10,101) (CIV(I),I=1,NTORD)
      WRITE(9,201) (CIV(I),I=1,NTORD)
 14   CALL MSTRCR(NVARS,NELMS,LIST,LPTR,TOTELM,NTREAT,NIP,
     *  IP,NRP,RP,NIF,NCF,TREAT)
      IF(TYPCEL.LE.2) CALL DICCT(NOF,NTORD,TYPCEL,IP1,CRV,
     *   LIST,LPTR,TOTELM,RCCT,TIEV)
      IF(TYPCEL.EQ.3) CALL TC3(NTREAT,LIST,LPTR,CRV,NTORD,
     *     IP1,NCOL,RCCT)
      IF(TYPCEL.EQ.4) CALL TC4(NTREAT,LIST,LPTR,CIV,
     *      NTORD,IP1,NCOL,RCCT)
      IF(TYPCEL.EQ.63) CALL MCOUNT(LIST,LPTR,CRV,RCCT,
     *      TREAT)
      IF(TYPCEL.EQ.73) CALL COCH
     * (NTREAT,NOF,LIST,LPTR,RP,NTORD,CRV,NROW,NCOL,RCCT)
      IF(TYPCEL.EQ.73) GO TO 99
      WRITE(9,301)
C     WRITE(NOF,301)
 301  FORMAT(1X,' CELL MATRIX:')
      DO 80 I=1,NROW
         LL=1+(I-1)*NCOL
         UL=I*NCOL
         WRITE (9,201) (RCCT(J),J=LL,UL)
C        WRITE (NOF,201) (RCCT(J),J=LL,UL)
 80   CONTINUE
 99   RETURN
      END
C-------------------------------------------------------------------
      SUBROUTINE COCH
     * (NTREAT,NCF,LIST,LPTR,RP,NTORD,CRV,NROW,NCOL,RCCT)
C        THIS FINDS WHICH DATA ELEMENTS ARE SUCCESSES,
```

```
C     BY COMPARING LIST WITH PT, COUNTS THE # OF SUCCESSES
C     PER ROW & COL, SENDS TOTS INTO RCCT,  INDEX 1-NROW=
C     SUCCESS PER ROW(BLOCK),NROW+1 TO NROW+NCOL=SUCCESS
C     PER COL(TREAT)
      REAL LIST(400),RP(50),CRV(50),PT,MULT,V
      INTEGER NTREAT,LPTR(17),NROW,NCOL,RCCT(50),N,IC,IR,
     *LL,UL,NT
      N=IFIX(FLOAT(NTORD)/2)
      IF (N.NE.NTREAT) WRITE (NOF,101)  N,NTREAT
 101  FORMAT(1X,
     * 'EXPECTED NUMBER OF TREATMENTS BASES ON NUMBER',
     * /'OF PARTITION VALUES WAS',I4,
     * ' WHICH DOES NOT AGREE WITH THE',
     * /'LISTED NUMBER OF TREATMENTS,',I4)
      NT=NROW+NCOL
      DO 5 I=1,NT
         RCCT(I)=0
 5    CONTINUE
      DO 10 I=1,N
         MULT=CRV((2*I))
         PT=CRV((2*I)-1)*MULT
         IC=I+NROW
         LL=LPTR(I)
         UL=LPTR(I+1)-1
         DO 20 J=LL,UL
               IR=J-LL+1
               V=LIST(J)*MULT
               IF(V.GT.PT) GO TO 20
                    RCCT(IR)=RCCT(IR)+1
                    RCCT(IC)=RCCT(IC)+1
 20      CONTINUE
 10   CONTINUE
      NROWP1=NROW+1
      WRITE (NOF,201) (RCCT(I),I=1,NROW)
      WRITE (NOF,202) (RCCT(J),J=NROWP1,NT)
 201  FORMAT(1X,'ROW TOTALS:'/20I3)
 202  FORMAT(1X,'COLUMN TOTALS:'/20I3)
      RETURN
      END
C-----------------------------------------------------------
      SUBROUTINE TC3
     * (NTREAT,LIST,LPTR,CRV,NTORD,NROW,NCOL,RCCT)
      INTEGER NTREAT,LPTR(17),NTORD,NROW,RCCT(50),CIV(50),
     *KEY2(200),NPAIR,ROWNR(50),COLNR(50),NCOL,KEY1(200)
      REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
      WRITE(9,101)
 101  FORMAT(1X,'INTO TC3')
      CALL PAIRST(VAL1,VAL2,KEY1,KEY2,LIST,LPTR,NPAIR)
      CALL TC3LN(1,CRV,VAL1,KEY1,NPAIR,NROW,NCOL,ROWNR)
      CALL TC3LN(0,CRV,VAL2,KEY2,NPAIR,NROW,NCOL,COLNR)
      CALL RCCTR(NROW,NCOL,NPAIR,ROWNR,COLNR,RCCT)
      RETURN
      END
C-----------------------------------------------------------
      SUBROUTINE TC4
     * (NTREAT,LIST,LPTR,CIV,NTORD,NROW,NCOL,RCCT)
      INTEGER NTREAT,LPTR(17),NTORD,NROW,NCOL,RCCT(50),
     *KEY2(200),NPAIR,ROWNR(50),COLNR(50),CIV(50),KEY1(200)
      REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
      CALL PAIRST(VAL1,VAL2,KEY1,KEY2,LIST,LPTR,NPAIR)
      CALL TC4LN(1,CIV,KEY1,NPAIR,NROW,NCOL,ROWNR)
      CALL TC4LN(0,CIV,KEY2,NPAIR,NROW,NCOL,COLNR)
      CALL RCCTR(NROW,NCOL,NPAIR,ROWNR,COLNR,RCCT)
      RETURN
      END
C---------------------------------------------------------
      SUBROUTINE PAIRST
```

```
     * (VAL1,VAL2,KEY1,KEY2,LIST,LPTR,NPAIR)
       INTEGER NTREAT,LPTR(17),NTORD,NROW,NCOL,RCCT(50),
      *KEY2(200),NPAIR,ROWNR(50),COLNR(50),CIV(50),
      *NRCELM,NTOP,ICIV,ICRV,NIN,I,J,JJ,K,RORC,KEY1(200)
       REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
       WRITE(9,101)
 101   FORMAT(1X,'INTO PAIRST')
       NPAIR=LPTR(2)-LPTR(1)
       DO 10 I=1,NPAIR
          J=I+NPAIR
          VAL1(I)=LIST(I)
          VAL2(I)=LIST(J)
          KEY1(I)=I
          KEY2(I)=I
 10    CONTINUE
       WRITE (9,305)
 305   FORMAT (1X,' VALUES OF VAL1,VAL2 BEFORE SORT ')
       WRITE(9,301) (VAL1(I),I=1,NPAIR)
       WRITE(9,301) (VAL2(I),I=1,NPAIR)
       CALL SHSORT(VAL1,KEY1,NPAIR)
       CALL SHSORT(VAL2,KEY2,NPAIR)
       WRITE(9,302)
 302   FORMAT (1X,' VALUES OF VAL1,VAL2 AFTER SORT ')
       WRITE(9,301) (VAL1(I),I=1,NPAIR)
       WRITE(9,301) (VAL2(I),I=1,NPAIR)
 301   FORMAT(7F10.3)
       RETURN
       END
C-----------------------------------------------------------------
       SUBROUTINE TC3LN
     * (RORC,CRV,VAL,KEY,NPAIR,NROW,NCOL,RORCNR)
       INTEGER NTREAT,LPTR(17),NTORD,NROW,NCOL,RCCT(50),
      *KEY2(200),NPAIR,ROWNR(50),COLNR(50),CIV(50),IK,
      *NRCELM,NTOP,ICIV,ICRV,NIN,I,J,JJ,K,RORC,RORCNR(50),
      *KEY1(200),KEY(200)
       REAL LIST(400),VAL(200),CRV(50)
C      WRITE (9,101)NPAIR
C101   FORMAT(1X,' INTO TC3LN ,NPAIR:',I5)
       IK=1
       IF(RORC.LT.1) GO TO 10
          ICRV=1
          K=0
          NTOP=NROW-1
          GO TO 20
 10    CONTINUE
          ICRV=NROW
          K=NROW-1
          NTOP=NROW+NCOL-2
 20    CONTINUE
       WRITE (9,203) IK,ICRV,K,NTOP,VAL(IK),CRV(ICRV)
 203   FORMAT(1X,'IK,ICRV,K,NTOP:',4I5,
     *    ' VAL(K),CRV(ICRV) ',2F10.3)
       IF ((ICRV.GT.NTOP).OR.(VAL(IK).LE.CRV(ICRV)))
     *    GO TO 12
          ICRV=ICRV+1
          GO TO 20
 12    RORCNR(KEY(IK))=ICRV-K
       WRITE (9,204) KEY(IK), RORCNR(KEY(IK))
 204   FORMAT (1X,'KEY(IK)',I5,' RORCNR(KEY(IK))',I5)
          IK=IK+1
          IF (IK.GT.NPAIR) GO TO 99
          GO TO 20
 99    RETURN
       END
C-----------------------------------------------------
       SUBROUTINE RCCTR(NROW,NCOL,NPAIR,ROWNR,COLNR,RCCT)
C      DIM
```

```
      INTEGER NTREAT,LPTR(17),NTORD,NROW,NCOL,RCCT(50),
     *KEY2(200),NPAIR,ROWNR(50),COLNR(50),CIV(50),CELLNR,
     *NRCELM,NTOP,ICIV,ICRV,NIN,I,J,JJ,K,KEY1(200)
      REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
      WRITE (9,101)
 101  FORMAT(1X,' INTO RCCTR')
      NRCELM=NROW*NCOL
      DO 10 I=1,NRCELM
         RCCT(I)=0
 10   CONTINUE
      DO 20 I=1,NPAIR
         CELLNR=(ROWNR(I)-1)*NCOL+COLNR(I)
         RCCT(CELLNR)=RCCT(CELLNR)+1
 20   CONTINUE
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE TC4LN(RORC,CIV,KEY,NPAIR,NROW,NCOL,RORCNR)
      INTEGER NTREAT,LPTR(17),NTORD,NROW,NCOL,RCCT(50),
     *KEY2(200),NPAIR,ROWNR(50),COLNR(50),CIV(50),
     *NRCELM,NTOP,ICIV,ICRV,NIN,I,J,JJ,K,RORC,RORCNR(50),
     *KEY1(200),KEY(200)
      REAL LIST(400),VAL1(200),VAL2(200),CRV(50)
      NRCELM=NROW*NCOL
      IF (RORC.LT.1) GO TO 10
         NTOP=NROW
         ICIV=1
         K=0
         GO TO 20
 10   NTOP=NROW+NCOL
         ICIV=NROW+1
         K=NROW
 20   J=0
      DO 30 I=ICIV,NTOP
         NIN=CIV(I)
         IF (NIN.EQ.0) GO TO 30
         DO 15 JJ=1,NIN
               J=J+1
               RORCNR(KEY(J))=I-K
 15      CONTINUE
 30   CONTINUE
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE DICOT(NOF,NTORD,TYPCEL,IP1,CRV,LIST,LPTR,
     *  TOTELM,RCCT,TIEV)
      INTEGER DICOP,LPTR(17),UL,I,NSUCC,L,D,KEY1(400),
     *  NTORD,TYPCEL,NPAIR,IP1,RCCT(50),NFAIL,I,TIEV(2),
     *  KEY2(400),TOTELM
      REAL DIVAL(200),CRV(200),LIST(TOTELM),PT,DIF,ADIF,
     *  EPSILN
      NSUCC=0
      TIEV(1)=0
      TIEV(2)=0
      EPSILN=0.0001
      NPAIR=LPTR(2)-1
      WRITE (9,206) (CRV(I),I=1,NTORD)
 206  FORMAT(1X,'CRV',7F10.3)
C----THIS SECTION IF DICOTOMIZE BY PARTITION----
      IF (TYPCEL.EQ.2) GO TO 40
         PT=CRV(1)
         DO 10 I=1,NPAIR
               IF (LIST(I).GT.PT) NSUCC=NSUCC+1
               DIF=ABS(LIST(I)-PT)
               IF (DIF.GT.EPSILN) GO TO 8
                    IF (LIST(I).LE.PT) TIEV(1)=TIEV(1)+1
                    IF (LIST(I).GT.PT) TIEV(2)=TIEV(2)+1
```

```
  8           WRITE (9,207) LIST(I),PT,NSUCC
 207          FORMAT(1X,'LIST(I),PT,NSUCC',2F10.3,I5)
 10      CONTINUE
         GO TO 99
C----THE FOLLOWING SECTION IF DICOTOMIZE BY LIST---
 40   CONTINUE
      DO 41 I=1,NPAIR
         KEY1(I)=I
 41   CONTINUE
      DO 42 I=1,NTORD
         KEY2(I)=I
         DIVAL(I)=CRV(I)
 42   CONTINUE
      CALL SHSORT(LIST,KEY1,NPAIR)
      CALL SHSORT(DIVAL,KEY2,NTORD)
      L=1
      D=1
 50   IF((L.GT.NPAIR).OR.(D.GT.NTORD)) GO TO 99
         DIF=LIST(L)-DIVAL(D)
         ADIF=ABS(DIF)
         WRITE (9,209) LIST(L),DIVAL(D),DIF,NSUCC
 209     FORMAT(1X,'LIST,DIVAL,DIF,NSUCC',3F10.5,I5)
         IF (ADIF.GT.EPSILN) GO TO 60
            NSUCC=NSUCC+1
            L=L+1
            GO TO 50
 60      IF(DIF.GT.0) D=D+1
         IF(DIF.LT.0) L=L+1
         GO TO 50
 99   WRITE (9,101) NSUCC,NPAIR
 101  FORMAT(1X,' # OF STANDARD SUCCESSES ',I5,' OF ',I5)
      IF (IP1.EQ.0) NSUCC=NPAIR-NSUCC
C***REFLECT THE # OF SUCCESSES IF LIST CONTAINS FAILURES***
C           OR VALUES ABOVE PARTITION ARE FAILURES
      WRITE (9,105) NSUCC,NPAIR
C     WRITE (NCF,105) NSUCC,NPAIR
 105  FORMAT(1X,' NUMBER OF SUCCESSES ',I5,' OF ',I5)
      NFAIL=NPAIR-NSUCC
      RCCT(1)=NSUCC
      RCCT(2)=NFAIL
      RETURN
      END
C---------------------------------------------------------------
      SUBROUTINE MCCUNT (LIST,LPTR,CRV,RCCT,TREAT)
C   THIS GETS COUNTS FOR FOUR CELLS OF THE MCNEMAR TEST:
C 2 SUCCESS,2 FAILURE, SUCCESS-FAILURE,FAILURE-SUCCESS
C  FBS=1.0 IF SUCCESS IN FIRST TREATMENT IS BELOW CRV(1),
C  =-1.0 IF SUCCESS IS ABOVE CRV(1)
      REAL LIST(400),CRV(50),FBS,SBS,FPV,SPV,F,S
      INTEGER LPTR(17),RCCT(4),N,K,I,LF,LS,TREAT(16)
      N=LPTR(2)-1
      FBS=CRV(3)
      SBS=CRV(4)
      FPV=CRV(1)*FBS
      SPV=CRV(2)*SBS
      LF=1+IFIX(CRV(5))
      LS=2-IFIX(CRV(5))
      DO 5 I=1,4
         RCCT(I)=0
5     CONTINUE
      DO 10 I=1,N
         K=1
         F=LIST(I)*FBS
         S=LIST(I+N)*SBS
         IF(F.GT.FPV) K=K+LF
         IF(S.GT.SPV) K=K+LS
         RCCT(K)=RCCT(K)+1
```

```
 10    CONTINUE
       IF (CRV(5).LT.0.5) GO TO 20
       WRITE (NOF,111) TREAT(1),TREAT(2)
       GO TO 30
 20    WRITE (NOF,111) TREAT(2),TREAT(1)
 111   FORMAT(1X,'TREATEMENTS:'/'  BEFORE',I5,'  AFTER',I5)
 30    RETURN
       END
C*****************************************************************
       SUBROUTINE MSTRDR(NVARS,NELMS,LIST,LPTR,TOTELM,
     * NTREAT,NIP,IP,NRP,RP,NIF,NOF,TREAT)
C      THIS SUBPROGRAM GROUP READS ALL OF THE DATA FILE EXCEPT
C  THE NAMES, AND LINES 4-7 OF THE OPTIONS FILE.  IT CHOOSES
C  THE DATA ELEMENTS OF THOSE VARIABLES INCLUDED IN THE TEST,
C  AND PUTS THEM INTO THE VECTOR LIST.  IT THEN REMOVES THE
C  DATA ELEMENTS IF THE USER INDICATED TO EXCLUDE PAIRS OR
C  BLOCKS. THE INTEGER VECTOR LPTR DIMENSIONS LIST; LPTR(1)
C  = 1; LPTR(I) = 1ST ELEM OF ITH VAR IN LIST. NTREAT =
C  # TREATMENTS/VARS IN TEST; TOTELM IS TOTAL ELEMS IN LIST
C  LPTR(NTREAT+1) = TOTELM+1.  NELMS IS INT VECTOR OF ELMS
C  PER VAR AFTER BLOCKS/PAIRS EXCLUDED; TREAT IS INT VECTOR
C  OF INDICES OF ORIGINAL DATA SET WHICH ARE INCLUDED IN TEST
C  IP IS INT VECTOR OF INT PARAMS; RP IS REAL PARAMS
C
C       SUBROUTINES
C  MSTRDR   READS THE FILES
C  SUBLST   PREPARES DATA ELEMENTS FOR REDUCTION
C  NULIST   INCLUDES ONLY TREAT SPECIFIED INTO LIST
C  EXBLCC   EXCLUDES PAIRS OR BLOCKS AS REQ
       INTEGER NIF,NFILE,IHOLD(16),NVARS,TOTELM,EXB(200),
     *NTREAT,IP(50),LPTR(17),NELMS(16),RNELMS(16),RNVARS,
     *DICOP,NSUCC,CELLS,NTEST,TREAT(16),REDELM,NRP
       REAL RP(10),LIST(400),DIVAL(200)
 101   FORMAT(16I5)
 102   FORMAT(8F10.3)
 201   FORMAT(1X,16I5)
 202   FORMAT(1X,8F10.3)
       DICOP=0
       CALL FNAME (1,NIF,NOF)
       CALL FNAME (2,NIF,9)
       CALL FNAME (2,NIF,NOF)
       READ(NIF,101) (IHOLD(I),I=1,16)
       NVARS=IHOLD(1)
       LPTR(1)=1
       TOTELM=0
       DO 10 I=1,NVARS
          IP1=I+1
          NELMS(I)=IHOLD(IP1)
          LPTR(IP1)=LPTR(I)+NELMS(I)
          TOTELM=TOTELM+NELMS(I)
 10    CONTINUE
       READ(NIF,102)(LIST(I),I=1,TOTELM)
       CALL SUBLST(LIST,LPTR,NVARS,NTREAT,TOTELM,TREAT)
       CALL RPARAM(NIP,NRP,IP,RP)
       WRITE (9,407)
 407   FORMAT(1X,' LISTPTR VECTOR AND LIST AFTER REDUCTION')
       WRITE(9,201) (LPTR(I),I=1,17)
       WRITE(9,202) (LIST(I),I=1,TOTELM)
       WRITE(9,303)
 303   FORMAT(1X,' NVARS,TOTELM,NTEST,NTREAT,NEXB,NIP,NRP')
       WRITE(9,201) NVARS,TOTELM,NTEST,NTREAT,NEXB,NIP,NRP
       RETURN
       END
C-----------------------------------------------------------------
       SUBROUTINE SUBLST(LIST,LPTR,NVARS,NTREAT,TOTELM,TREAT)
C  THIS SUBROUTINE READS THE VALUES FOR THE INCLUDED TREATS,
C AND EXCLUDED BLOCKS, IF ANY, AND REDUCES THE ORIGINAL
```

```
C DATA SET OF THE DATA FILE INTO THE DATA LIST FOR THE TEST
C SUBROUTINE NULIST INCLUDES TREATMENTS, EXBLOC EXCLUDES
C         TREATMENTS
      REAL LIST(400)
      INTEGER LPTR(17),NVARS,NTREAT,TOTELM,TREAT(16),NEXB,
     *RNELMS(16),RNVARS,EXB(200)
 101  FORMAT(16I5)
 102  FORMAT(8F10.3)
 201  FORMAT(1X,16I5)
 202  FORMAT(1X,8F10.3)
      DO 8 I=1,NVARS
         TREAT(I)=0
 8    CONTINUE
      READ(10,101) NTREAT
      WRITE (9,301) NTREAT
 301  FORMAT(1X,'NUMBER OF TREATMENTS',I5,' INDICES')
      READ(10,101) (TREAT(I),I=1,NTREAT)
      WRITE(9,201) (TREAT(I),I=1,NTREAT)
      READ(10,101) NEXB
      WRITE (9,302) NEXB
 302  FORMAT(1X,'NUMBER OF EX BLOCKS',I5,' INDICES')
      IF (NEXB.GE.1) READ(10,101) (EXB(I),I=1,NEXB)
      IF (NEXB.GE.1) WRITE(9,201) (EXB(I),I=1,NEXB)
      CALL NULIST
     *    (NELMS,TOTELM,LIST,NTREAT,TREAT,LPTR,RNVARS)
      IF (NEXB.GE.1) CALL EXBLOC(LPTR,TOTELM,LIST,NTREAT,
     * NELMS,NEXB,EXB,RNELMS)
      RETURN
      END
C------------------------------------------------------------------
      SUBROUTINE NULIST
     *    (NELMS,TOTELM,LIST,NTREAT,TREAT,LPTR,RNVARS)
      INTEGER NIF,NFILE,IHCLD(16),TOTELM,NTEST,EXB(200),
     *NTREAT,IP(50),NRP,LPTR(17),NELMS(16),NULPTR(17),KP1,
     *RNVARS,NCF,II,UL,LL,NVARS,TREAT(16)
      REAL RP(10),LIST(400)
      IF(NVARS.EQ.NTREAT) GO TO 98
      DO 5 K=1,17
         NULPTR(K)=0
 5    CONTINUE
      I=0
      NULPTR(1)=1
      DO 10 II=1,NTREAT
         IIP1=II+1
         K=TREAT(II)
         KP1=K+1
         LL=LPTR(K)
         UL=LPTR(KP1)-1
         DO 15 J=LL,UL
               I=I+1
               LIST(I)=LIST(J)
 15      CONTINUE
      NULPTR(IIP1)=UL+1-LL+NULPTR(II)
 10   CONTINUE
      TOTELM=I
      WRITE(9,408) (NULPTR(I),I=1,17)
 408  FORMAT(1X,'NULPTR',/16I5)
      LPTR(1)=1
      DO 20 K=1,17
         LPTR(K)=NULPTR(K)
 20   CONTINUE
 98   RNVARS=NTREAT
      RETURN
      END
C-------------------------------------------------------------------
      SUBROUTINE EXBLOC
     *     (LPTR,TOTELM,LIST,NTREAT,NELMS,NEXB,EXB,RNELMS)
```

```
      INTEGER NIF,NFILE,IHOLD(16),TOTELM,NTEST,EXB(200),I,
     *NTREAT,IP(50),NRP,LPTR(17),NELMS(16),EXI(200),NEXI,K,
     *RNELMS(16),TEP1,NCF,LL,UL,NVARS,TREAT(16),EXTHIS,J
      REAL RP(10),LIST(400)
      IF (NEXB.EQ.0) GO TO 97
      K=0
C GET INDICES OF LIST ELEMENTS TO BE EXCLUDED
      LPTR(1)=1
      DO 10 I=1,NTREAT
         DO 12 J=1,NEXB
            K=K+1
            EXI(K)=LPTR(I)+EXB(J)-1
 12      CONTINUE
 10   CONTINUE
      NEXI=K+1
      EXI(NEXI)=17
      K=1
      J=0
      TEP1=TOTELM+1
C RELOAD LIST, EXLUDING ELEMENTS AS INDICATED BY EXI
      DO 20 I=1,TEP1
         EXTHIS=EXI(K)
         IF (I.LT.EXTHIS) J=J+1
         IF (I.LT.EXTHIS) LIST(J)=LIST(I)
         IF (I.GE.EXTHIS) K=K+1
 20   CONTINUE
C CHANGE LIST POINTER,NUMBER OF ELEMENTS
      LPTR(1)=1
      TOTELM=TOTELM-NEXB*NTREAT
      DO 25 I=1,NTREAT
         RNELMS(I)=NELMS(TREAT(I))-NEXB
 25   CONTINUE
      NTP1=NTREAT+1
      DO 30 I=2,NTP1
         LPTR(I)=LPTR(I)-(I-1)*NEXB
 30   CONTINUE
      LL=NTREAT+2
      DO 40 K=LL,17
         LPTR(K)=0
 40   CONTINUE
      GO TO 99
 97   DO 98 I=1,NTREAT
         RNELMS(I)=NELMS(TREAT(I))
 98   CONTINUE
 99   RETURN
      END
C**********************************************************************
C THIS SECTION CONTAINS THE SUBROUTINES CALLED BY MORE THAN
C ONE TEST SUBROUTINE. THEY APPEAR IN ALPHABETICAL ORDER:
C EXPF, FINDS, FINDT, MUVAR, NORMF, NORMV, PARCOM, RANQUE
C RHOER, RNQ, RPARAM
C-----------------------------------------------------------------------
      SUBROUTINE EXPF(LIST,N,RP,F,NOF)
C        THIS SUBROUTINE FINDS THE THEORETICAL EXPONENTIAL
C     DISTIBUTION F, WHERE L=RP(1)=SCALE PARAMETER LAMBDA
      REAL A,L,X,LIST(400),RP(50),F(400)
      INTEGER I
      L=RP(1)
      DO 10 I=1,N
         X=LIST(I)
         IF(X.LE.0) F(I)=0
         IF(X.GT.0) F(I)=1.0-EXP(-(L*X))
 10   CONTINUE
      WRITE(NOF,101) L
 101  FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTUION ',
     * 'EXPONENTIAL'/1X,'WITH PARAMETER LAMBDA=',F10.4)
      RETURN
```

```
      END
C----------------------------------------------------------------------
      SUBROUTINE FINDS(LIST,LPTR,S,K)
C  THIS SUBROUTINE FINDS THE DIFFERENCES BETWEEN TWO
C EMPIRICAL CDFS FOR THE TWO VECTORS CONTAINED IN LIST
C 1ST VAR X HAS # ELMS M AND COUNTER I; 2D VAR Y HAS # ELMS
C N AND COUNTER J; E CDF ARE (I-1)/M AND (J-1)/N; STEP
C COUNTER OF SMALLER VAR UNTIL N+1 OR M+1, THEN INCR OTHER
      REAL  EPSILN,FN,FM,X(100),Y(100),DIF,ADIF,S(200),
     *   ATMAX,ATMIN,TSUP,PDIF,LIST(400),RP(50),TMAX,TMIN
      INTEGER  I,J,K,L,M,N,KEYI(100),KEYJ(100),JJ,IP(50),
     *   LPTR(17),NCF,TREAT(17),II
      EPSILN=0.0001
      M=LPTR(2)-1
      N=LPTR(3)-LPTR(2)
      FN=FLOAT(N)
      FM=FLOAT(M)
      DO 10 I=1,M
         KEYI(I)=I
         X(I)=LIST(I)
 10   CONTINUE
      CALL  SHSORT(X,KEYI,M)
      DO 12 J=1,N
         KEYJ(J)=J
         Y(J)=LIST((M+J))
 12   CONTINUE
      CALL  SHSORT(Y,KEYJ,N)
      K=0
      L=0
      I=1
      J=1
 20   CONTINUE
      IF ((I.GT.M).AND.(J.GT.N)) L=1
 101  FORMAT(1X,'ITERATION #,I,J,STOP?',4I5,' DIF:',F10.4)
 111  FORMAT(1X,'S(K):',F10.4)
      II=I
      IF(I.GT.M) II=M
      JJ=J
      IF(J.GT.N) JJ=N
      DIF=X(II)-Y(JJ)
      ADIF=ABS(DIF)
      IF(ADIF.GT.EPSILN) GO TO 25
         IF(I.LE.M) I=I+1
         PDIF=FLOAT(I-1)/FM-FLOAT(J-1)/FN
         K=K+1
         S(K)=PDIF
         WRITE (9,101) K,I,J,L,DIF
         IF (K.GT.0) WRITE (9,111) S(K)
         IF(J.LE.N) J=J+1
         K=K+1
         S(K)=PDIF
         WRITE (9,101) K,I,J,L,DIF
         IF (K.GT.0) WRITE (9,111) S(K)
         IF(L.EQ.1) GO TO 30
         GO TO 20
 25   CONTINUE
         IF((DIF.LT.0).AND.(I.LE.M)) I=I+1
         IF((DIF.GT.0).AND.(J.LE.N)) J=J+1
         K=K+1
         S(K)=FLOAT(I-1)/FM-FLOAT(J-1)/FN
         WRITE (9,101) K,I,J,L,DIF
         IF (K.GT.0) WRITE (9,111) S(K)
         IF((DIF.LT.0).AND.(I.GT.M).AND.(J.LE.N)) J=J+1
         IF((DIF.GT.0).AND.(J.GT.N).AND.(I.LE.M)) I=I+1
         IF(L.LT.1) GO TO 20
 30   RETURN
      END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE FINDT(S,F,N,TSUP,TMIN,TMAX,NOF)
C     THIS FINDS THE DIFFERENCE IN CDF'S OF THEORETICAL
C DISTRIBUTION F AND EMPIRICAL DIST S AT THE X'S DETERMINED
C BY THE VALUES OF THE SAMPLE DATA VARIABLE ELEMENTS.  THE
C MAX ABSOLUTE DIFFERENC IS TSUP. T IS VECTOR OF DIFFERENCE:
      REAL S(400),F(400),TSUP,ATMIN,ATMAX,TMIN,TMAX,T(800)
      INTEGER N,I,II,NOF
      NP1=N+1
      T(1)=F(1)
      T(2)=F(1)-S(1)
      DO 10 II=2,N
         I=II-1
         IT2=(II*2)
         IT1=IT2-1
         T(IT1)=F(II)-S(I)
         T(IT2)=F(II)-S(II)
 10   CONTINUE
      TMIN=2.0
      TMAX=-2.0
      DO 12 J=1,IT2
         IF(T(J).GT.TMAX) TMAX=T(J)
         IF(T(J).LT.TMIN) TMIN=T(J)
 12   CONTINUE
      ATMAX=ABS(TMAX)
      ATMIN=ABS(TMIN)
      TSUP=ATMIN
      IF(ATMAX.GT.TSUP) TSUP=ATMAX
      WRITE(NOF,101) TMAX,TMIN,TSUP
 101  FORMAT(1X,'MAXIMUM DIFFERENCE, F(X)-S(X),',F10.4/1X,
     *' MINIMUM DIFFERENCE:',F10.4/1X,
     *'LARGEST ABSOLUTE DIFFERENCE:',F10.4)
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE MUVAR(LIST,N,MU,VAR)
C             THIS SUBROUTINE FINDS THE SAMPLE MEAN AND VARIANCE
      REAL LIST(400),MU,VAR,XBAR,SUMSQ
      INTEGER N
      XBAR=0.0
      SUMSQ=0.0
      DO 10 I=1,N
         XBAR=XBAR+LIST(I)
 10   CONTINUE
      MU=XBAR/FLOAT(N)
      DO 20 I=1,N
         SUMSQ=SUMSQ+(LIST(I)-MU)**2
 20   CONTINUE
      VAR=SUMSQ/FLOAT(N-1)
      WRITE (9,101) MU,VAR
      WRITE (NOF,101) MU,VAR
 101  FORMAT(1X,' ESTIMATED MU',F10.3,'  VARIANCE',F10.3)
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE NORMF(LIST,N,RP,F,NOF)
C             THIS SUBROUTINE FINDS THE THEORETICAL NORMAL
C     DISTIBUTION F, WHERE MU=RP(1),SIGMA SQUARED=RP(2)
      REAL PX,MU,SIGSQ,X,LIST(400),RP(50),F(400)
      INTEGER I
      MU=RP(1)
      SIGSQ=RP(2)
      DO 10 I=1,N
         X=LIST(I)
         Z=(X-MU)/SIGSQ
         CALL NORMV(1,Z,PX)
         F(I)=PX
```

```
 10   CONTINUE
      WRITE(NOF,101) MU,SIGSQ
 101  FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTUION NORMAL'/1X,
     *'WITH PARAMS MU=',F10.4,', AND SIGMA SQUARED=',F10.4)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE NORMV(OPTION,Z,P)
C IF OPTION=1, THIS TAKES A Z(STANDARD NORMAL) AND RETURNS
C A P, IF OPTION=2, TAKES A P AND RETURNS A Z
C NUMERICAL INTEGRATION, TRAPAZOIDAL, STEP=0.1 STD DEV
C INITIALIZE BOTH Z AND P IN CALLING SUBROUTINE
      REAL Z,P,PI,CONST,OLD,NEW,ZZ,PP,CUMP,X,DELTP,EXCESP,
     *DELX
      INTEGER OPTION,I,J,UL,ONE
      PI=3.1415927
      CUMP=0
      CONST=1.0/(2*PI)**0.5
      OLD=CONST
      IF (OPTION.EQ.2) GO TO 50
         ZZ=ABS(Z)
         UL=IFIX(ZZ*10.0)
         DO 10 I=1,UL
               X=FLOAT(I)*0.1
               NEW=CONST*EXP(-0.5*X*X)
               CUMP=CUMP+0.05*(OLD+NEW)
               OLD=NEW
               IF (CUMP.GT.0.48) GO TO 12
               WRITE (9,104) I,NEW,CUMP,X
 104           FORMAT(1X,'I',I5,' NEW,CUMP,X',3F7.3)
 10      CONTINUE
         DELX=ZZ-X
         NEW=CONST*EXP(-0.5*ZZ*ZZ)
         CUMP=CUMP+0.5*DELX*(OLD+NEW)
         WRITE (9,104) I,NEW,CUMP,X
 12      P=CUMP+0.5
         IF (Z.LT.0.0) P=0.5-CUMP
         GO TO 95
 50   CONTINUE
         PP=P-0.5
         X=0.0
         IF (P.LT.0.5) PP=0.5-P
 60      X=X+0.1
               NEW=CONST*EXP(-0.5*X*X)
               DELTP=0.05*(OLD+NEW)
               CUMP=CUMP+DELTP
               OLD=NEW
               WRITE(9,109) X,NEW,CUMP
 109           FORMAT(1X,' X,NEW,CUMP',3F7.3)
         IF (CUMP.LT.PP) GO TO 60
         EXCESP=CUMP-PP
         ZZ=X-0.1*EXCESP/DELTP
         Z=ZZ
         IF (P.LT.0.5) Z=-ZZ
 95   WRITE (9,101) OPTION,P,Z
 101  FORMAT(1X,' OPTION, PROB, VAL OF Z:',I5,2F10.3)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE PARCOM(V,T,S,NOF)
C  THIS FIND THE DIFFERENCES BETWEEN THE ITH AND JTH ELEMENT
C OF VECTOR V, FOR ALL I<>J, 1,...,T; PRINTS THE DIFFERENCES
C AND THE SUPPLIED STANDARD DEVIATION S
      REAL V(17),S
      INTEGER NOF,T,I,J,TM1,IP1
      WRITE(NOF,101) S
 101  FORMAT(9X,'//PAIRWISE COMPARISONS//'/1X,'IF AND ONLY',
```

```
     *' IF THE NULL HYPOTHESIS IS REJECTED'/1X,' THESE',
     *' COMPARISONS HAVE MEANING.  COMPARE THE',/1X,
     *' DESIRED QUANTILE OF THE T DISTRIBUTION'/1X,
     *' TIMES THE STANDARD DEVIATION ',F10.4,/1X,
     *' WITH THE FOLLOWING ABSOLUTE DIFFERENCES IN THE'/1X,
     *' SUMS OF ROW RANKS:'//1X,
     *'   I     J      |SR(I)-SR(J)|    '/)
      TM1=T-1
      DO 10 I=1,TM1
         IP1=I+1
         DO 20 J=IP1,T
            X=V(I)-V(J)
            AX=ABS(X)
            WRITE(NOF,102) I,J,AX
102         FORMAT(1X,2I5,3X,F10.4)
20       CONTINUE
10    CONTINUE
      RETURN
      END
C-------------------------------------------------------------------
      SUBROUTINE RANQUE(NVALS,VALS,SVALS,RANK,Q,KEY,NTREAT,
     *  NBLOCK,OPTION)
      DIMENSION VALS(NVALS),Q(200),RANK(NVALS),RANQ(400),
     *SVALS(NVALS),KEY(NVALS),JJV(400),RANGE(200),QKEY(200)
      INTEGER KEY,NEWTIE,OLDTIE,LL,NTREAT,NBLOCK,OPTION,
     *  JJV,NL,JJ,QKEY
      REAL DIF,VALS,SVALS,RANK1,RANK,AVE,EPSILN,RANQ,Q,RANGE
C
C        THESE TWO SUBROUTINES RANK VALUES
C
C Q         THE RANK OF RANGES WITHIN BLOCKS
C DIF       DIFFERENCE BETWEEN KTH AND K+1TH SORTED VALUE
C EPSILN TOLERANCE OF ROUNDING ERROR, I.E.,CLOSE ENOUGH TO
C              SAY TWO VALS ARE TIED
C NEWTIE    KTH AND K+1TH SORTED VALUES ARE TIED (LOGICAL INT)
C OLDTIE    K-1TH AND KTH SORTED VALUES ARE TIED (LOGICAL INT)
C KEY       ORIGINAL INDEX OF VALS,E.G.,KEY(1) HAS ORIG INDEX
C           OF SMALLEST VAL,KEY(NVALS) HAS INDEX OF LARGEST VA
C RANK      RANK OF THE UNSORTED (ORIGINAL) VALUES
C LL        LOWEST INDEX OF TIED SORTED VALUES
C AVE       AVERAGE OF LL & HIGHEST INDEX OF TIED SORTED VALUE
C RANQ      IF OPTION=1,TOTAL SET OF RANKS, =2 RANKS W/I BLOCK
C           =3 RANKS W/I BLOCK PLUS RANKS OF RANGES OF BLOCKS
C
      IF (OPTION.GE.2) GO TO 51
      NL=NVALS
      DO 10 I=1,NVALS
         SVALS(I)=VALS(I)
         KEY(I)=I
 10   CONTINUE
      CALL RNQ (SVALS,KEY,NL,RANQ)
      DO 15 I=1,NVALS
         RANK(I)=RANQ(I)
 15   CONTINUE
      GO TO 95
 51   NL=NTREAT
      DO 20 I=1,NBLOCK
         QKEY(I)=I
         DO 25 J=1,NL
            JJ=I+(J-1)*NBLOCK
            SVALS(J)=VALS(JJ)
            JJV(J)=JJ
            KEY(J)=J
 25      CONTINUE
         CALL RNQ (SVALS,KEY,NL,RANQ)
         RANGE(I)=SVALS(NL)-SVALS(1)
         WRITE (9,107) (RANQ(K),K=1,NL)
```

```
 107     FORMAT (1X, 7F10.3)
         DO 30 J=1,NL
            RANK(JJV(J))=RANQ(J)
 30      CONTINUE
 20   CONTINUE
      IF (OPTION.EQ.3) CALL RNQ(RANGE,QKEY,NBLOCK,Q)
 95   RETURN
      END
C----------------------------------------------------------------------
      SUBROUTINE RHOER(X,NPAIR,RHO,B1,B0,OPTION)
C  THIS SUBROUTINE TAKES X, A VECTOR COMPOSED OF TWO EQUAL
C LENGTHED VARIABLES, AND FINDS THE CORRELATION COEFFICIENT
C RHO (PEARSON'S: COVARIANCE/STDEV1*STDEV2)
      REAL X(400),RHO,SSA,SSB,FN,AB,DA,DB,AMEAN,BMEAN,
     *B1,B0,RHO,DENOM
      INTEGER NPAIR,OPTION
      FN=FLOAT(NPAIR)
      AMEAN=0.0
      BMEAN=0.0
      SSA=0.0
      SSB=0.0
      AB=0.0
      B1=0.0
      B0=0.0
      DO 10 I=1,NPAIR
         AMEAN=AMEAN+X(I)
         BMEAN=BMEAN+X((I+NPAIR))
 10   CONTINUE
      AMEAN=AMEAN/FN
      BMEAN=BMEAN/FN
      DO 20 I=1,NPAIR
         DA=X(I)-AMEAN
         DB=X((I+NPAIR))-BMEAN
         AB=AB+(DA*DB)
         SSA=SSA+(DA*DA)
         SSB=SSB+(DB*DB)
 20   CONTINUE
      DENOM=(SSA**0.5)*(SSB**0.5)
      RHO=AB/DENOM
      IF (OPTION.EQ.1) B1=AB/SSB
      IF (OPTION.EQ.1) B0=AMEAN-B1*BMEAN
      IF (OPTION.EQ.2) B1=AB/SSA
      IF (OPTION.EQ.2) B0=BMEAN-B1*AMEAN
      WRITE(9,122) RHO,B0,B1
 122  FORMAT(1X,' RHO,B0,B1:',3F10.4)
      RETURN
      END
C----------------------------------------------------------------------
      SUBROUTINE RNQ (SVALS,KEY,NL,RANQ)
C     THIS SUBROUTINE SORTS AND RANKS A VECTOR FROM RANQUE
C OR ANY OTHER SUBROUTINE;SVALS ARE UNSORTED VALUES WHICH
C WHICH ARE RETURNED SORTED AND DETERMINE RANKS; KEY IS THE
C VECTOR 1,...,NL WHERE NL IS # ELMS IN SVAL; RANQ IS RANKS
C OLDTIE,NEWTIE ARE LOGICALS TO START AND END TIED VALUES
      REAL SVALS(NL),RANQ(NL),EPSILN,
     *AVE,DIF
      INTEGER NL,I,LL,IP1,NEWTIE,OLDTIE,KEY(NL),K
      EPSILN=0.0001
      WRITE(9,222)
 222  FORMAT(10X,'**** START RANKING IN RNQ ****')
      CALL SHSORT(SVALS,KEY,NL)
      OLDTIE=0
      DO 10 I=1,NL
         RANQ(I)=FLOAT(I)
 10   CONTINUE
      DO 30 I=1,NL
         NEWTIE=0
```

```
          K=KEY(I)
          RANQ(K)=FLOAT(I)
C         IF (I.GT.1) WRITE (9,104) I,K,RANQ(K),RANQ(I)
C104      FORMAT (1X,'I,K ',2I5,' RANQ(K) ',F10.3,
C      *' RANQ(I)',F10.3)
          IP1=I+1
          DIF=SVALS(IP1)-SVALS(I)
          IF (I.EQ.NL) DIF=1.0
          IF (DIF.LT.EPSILN) NEWTIE=1
          IF (NEWTIE.EQ.OLDTIE) GO TO 29
          IF (NEWTIE.LT.OLDTIE) GO TO 15
              LL=I
              GO TO 29
 15       AVE=0.5*(FLOAT(I+LL))
          DO 20 J=LL,I
              RANQ(KEY(J))=AVE
 20       CONTINUE
 29     OLDTIE=NEWTIE
C         WRITE (9,107) (RANQ(K),K=1,NL)
 30     CONTINUE
          WRITE(9,121)
 121      FORMAT(1X,' SORTED VALUES')
          WRITE (9,107) (SVALS(K),K=1,NL)
          WRITE (9,122)
 122      FORMAT(1X,' RANKS OF VALUES ')
          WRITE (9,107) (RANQ(K),K=1,NL)
C         WRITE (9,108) (KEY(K),K=1,NL)
 107      FORMAT (1X, 7F10.3)
C108      FORMAT (1X, 7I10)
        RETURN
        END
C-----------------------------------------------------------------
        SUBROUTINE RPARAM(NIP,NRP,IP,RP)
C THIS READS THE INTEGER AND REAL PARAMETERS OF THE OPTIONS
C FILE FOR EITHER MSTROR OR CELLER
        INTEGER NIP,NRP,IP(50)
        REAL RP(10)
 101    FORMAT(16I5)
 102    FORMAT(8F10.3)
        READ(10,101) NIP
        IF (NIP.GE.1) READ(10,101) (IP(I),I=1,NIP)
        READ(10,101) NRP
        IF (NRP.GE.1) READ(10,102) (RP(I),I=1,NRP)
        RETURN
        END
C-----------------------------------------------------------------
        SUBROUTINE THYPA(X,NOF)
C THIS WRITES THE RESULT FOR A KOLMOGOROV, LILLIEFORS
C TEST TO THE OUTPUT FILE WHEN THE HYP=1
        WRITE(NOF,101) X
 101    FORMAT(1X,' THE NULL HYPOTHESIS: F*(X)=S(X)'/1X,'THE',
     *' TEST STATISTIC IS LARGER OF THE ABSOLUTES VALUES'/
     *3X,'OF THE TWO DIFFERENCES:'/1X,F10.4)
        RETURN
        END
C-----------------------------------------------------------------
        SUBROUTINE THYPB(X,NOF)
C THIS WRITES THE RESULT FOR A KOLMOGOROV, LILLIEFORS
C TEST TO THE OUTPUT FILE WHEN THE HYP=2
        WRITE(NOF,101) X
 101    FORMAT(1X,'THE NULL HYPOTHESIS: F*(X)<=S(X)'/1X,'THE',
     *' TEST STATISTIC IS LARGEST DIFFERENCE '/
     *3X,'OF F*(XI)-S(XI):'/1X,F10.4)
        RETURN
        END
C-----------------------------------------------------------------
        SUBROUTINE THYPC(X,NOF)
```

```
C THIS WRITES THE RESULT FOR A KOLMOGOROV, LILLIEFORS
C TEST TO THE OUTPUT FILE WHEN THE HYP=3
      WRITE(NOF,101) X
 101  FORMAT(1X,'THE NULL HYPOTHESIS:  F*(X)=>S(X)'/1X,'THE',
     *' TEST STATISTIC IS SMALLEST (OR MOST NEGATIVE)'/
     *3X,'OF F*(XI)-S(XI): '/1X,F10.4)
      RETURN
      END
C****************************************************
C  THIS SECTION OF THE PROGRAM CONTAINS THE TEST SUBROUTINES
CIN THE ORDER OF THEIR TEST NUMBER; TESTS ARE SEPARATED BY
C -/-/-/
      SUBROUTINE BINTST(NOF,RCCT,RP,IP,ALPHAT,CILL,CIUL)
C THIS MASTER SUBROUTINE FINDS ALPHA HAT OF A BINOMIAL TEST
C BASED ON THE TOTAL NUMBER N, THE NUMBER OF SUCCESSES T,
C THE HYPOTHESIZED PROBABILITY PSTAR,AND FORM OF HYPOTHESIS.
C THE NULL HYPOTHESIS IS EVALUATED. IT THEN FINDS LOWER AND
C UPPER CONFIDENCE INTERVAL BOUNDS (CILL,CIUL) FOR ACTUAL
C PROBABILITY BASED ON THE NR OF SUCCESES AND FAILURES.
C IF N<20, EXACT BINOMIAL COMPUTATIONS WILL BE PERFORMED,
C OTHERWISE THE NORMAL APPROXIMATION WILL BE USED.
       INTEGER NOF,IP(50),RCCT(50),N,T,HYP
       REAL ALPHA,ALPHAT,CIUL,CILL,RP(50),PSTAR,ALPHYP,
     * NKV(21)
      T=RCCT(1)
      N=RCCT(2)+T
      PSTAR=RP(1)
      ALPHYP=1.0-RP(2)
      HYP=IP(1)
      WRITE (NOF,201) N,T,PSTAR,ALPHYP
 201  FORMAT(10X,'BINOMIAL TEST:'/1X,'OF',I5,' ELEMENTS ',
     * 'OF DATA SET',I5,' WERE SUCCESSES'/1X,
     * ' THE HYPOTHESISED P STAR IS:',F10.5/
     *1X,' THE LEVEL OF SIGNIFICANCE IS:',F10.5)
      IF (HYP.EQ.1) WRITE (NOF,301)
 301  FORMAT(1X,
     *' THE NULL HYPOTHESIS IS P=P STAR, ALT IS P<>P STAR')
      IF (HYP.EQ.2) WRITE (NOF,302)
 302  FORMAT(1X,
     *' THE NULL HYPOTHESIS IS P<=P STAR, ALT IS P>P STAR')
      IF (HYP.EQ.3) WRITE (NOF,303)
 303  FORMAT(1X,
     *'THE NULL HYPOTHESIS IS P>=P STAR, ALT IS P<P STAR')
      IF (N.GT.20) GO TO 30
         CALL BINH2C(N,T,PSTAR,HYP,ALPHAT,NKV)
         CALL CIBINC(N,T,ALPHYP,NKV,CILL,CIUL)
         GO TO 99
 30   CONTINUE
      CALL BINORM(N,T,PSTAR,HYP,ALPHAT,ALPHYP,CILL,CIUL)
 99   CONTINUE
      WRITE (9,101) ALPHAT,CILL,CIUL
      WRITE (NOF,101) ALPHAT,CILL,CIUL
 101  FORMAT(1X,' ALPHA HAT, CI LOWER, UPPER',3F10.3)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE BINORM
     *      (N,T,PSTAR,HYP,ALPHAT,ALPHYP,CILL,CIUL)
C THIS SUBROUTINE FINDS THE ALPHA HAT OF A BINOMIAL DIST
C   AND THE CONFIDENCE LIMITS OF THE SAMPLES' PROBABLITY
C   CALLING THE SUBROUTINE NORMV
C ALPHYP  THE ALPHA SPECIFIED IN TEST 1.0-LEVEL OF SIGNIF
C ALPHAT  THE ALPHA RETURNED BY THE NORMV SUBROUTINE BASED
C           ON THE NUMBER OF SUCCESSES AND TOTAL SAMPLE
C ZHAT    THE EMPIRICAL Z = STANDARD NORMAL N SUCC MINUS
C           E(N SUCCESS)=(N*PSTAR),DIVIDED BY STD DEV (NPQ)
C ZHYP     Z RETURNED FROM NORMV BASED ON ALPHA* OF HYP
```

```
      INTEGER N,T,HYP
      REAL ZHAT,Z,ALPH,ALPHAT,CILL,CIUL,NPST,STDEV,ZHYP
      NPST=FLOAT(N)*PSTAR
      ZHAT=(FLOAT(T)-NPST)/((NPST*(1.0-PSTAR))**0.5)
      CALL NORMV(1,ZHAT,ALPHAT)
      IF (HYP.GT.1) GO TO 20
         IF (ZHAT.GT.0) ALPHAT=(1.0-ALPHAT)*2.0
         IF (ZHAT.LE.0) ALPHAT=ALPHAT*2.0
         GO TO 30
 20   IF (HYP.EQ.2) ALPHAT=1.0-ALPHAT
 30   STDEV=(FLOAT(T*(N-T))/((FLOAT(N))**3))**0.5
      CALL NORMV(2,ZHYP,ALPHYP)
      TDIVN=FLOAT(T)/FLOAT(N)
      CILL=TDIVN+ZHYP*STDEV
      CIUL=TDIVN-ZHYP*STDEV
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE NCHUZK(N,NKV)
C   THIS SOUBROUTINE PRODUCES THE NKV VECTOR OF N CHOOSE K
C     FOR K=0,... THE INDICES OF NKV ARE ONE
C     HIGHER THAN THE CORRESPONDING K,E.G.,I=1,...,N+1
      REAL NUM,DENOM,NKV(21)
      INTEGER UL,MED,NP1,IP1,NP1MI,N,I
      NP1=N+1
      UL=IFIX (N*0.5)
      MED=IFIX (NP1*0.5)
      NKV(1)=1
      NKV(NP1)=1
      NUM=1.0
      DENOM=1.0
      DO 10 I=1,UL
         IP1=I+1
         NP1MI=NP1-I
         NUM=NUM*FLOAT(NP1MI)
         DENOM=DENOM*FLOAT(I)
         NKV(IP1)=NUM/DENOM
         IF(I.LT.MED) NKV(NP1MI)=NKV(IP1)
 10   CONTINUE
      WRITE (9,101)
 101  FORMAT (1X,' N CHOOSE K VECTOR')
      WRITE (9,102) (NKV(I),I=1,NP1)
 102  FORMAT(8F10.3)
      RETURN
      END
C-------------------------------------------------------------------
      SUBROUTINE BINPMF(P,N,NKV,VECLIM,PQV,PMF,CUMPR)
C  THIS SUBROUTINE RETURNS THE VECTOR PQV CORRESPONDING TO
C P**K TIMES Q**(N-K) FOR K=0,...,VECLIM, AND PART OR ALL
C OF PMF FOR BINOMIAL. LET VECLIM = N TO PRODUCE ENTIRE PMF.
C FOR LINESEARCH, IT WILL BE LESS. VECTOR NKV FROM NCHUZK
C CUMPR IS CUMULATIVE PROBABILITY
      INTEGER I,VECLIM,N,K,UL
      REAL P,Q,PQV(21),PV(21),QV(21),PHOLD,QHOLD,NKV(21),
     *CUMPR,DIVER,PQ,PMF(21)
      CUMPR=0.0
      Q=1.0-P
      UL=VECLIM+1
      NP1=N+1
      DIVER=P/Q
      WRITE(9,103) P,Q,PQ,DIVER,VECLIM
 103  FORMAT(1X,'P,Q,PQ,DIVER,',4F10.7,'VECLIM',I5)
      PQ=Q**N
      DO 20 I=1,UL
         PQV(I)=PQ
         PQ=PQ*DIVER
         PMF(I)=PQV(I)*NKV(I)
```

```
         CUMPR=CUMPR+PMF(I)
 20   CONTINUE
      WRITE (9,101) VECLIM, P,CUMPR
 101  FORMAT (1X,'VECLIM',I5,
    *     ' P',F8.3,' CUM PROB:',F8.3,' P Q VECTOR:')
      WRITE (9,102) (PQV(I),I=1,NP1)
 102  FORMAT(8F10.3)
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BINH20(N,T,PSTAR,HYP,ALPHAT,NKV)
C     THIS FIGURES THE ALPHA HAT VALUE, AND RETURNS THAT
C VALUE AND THE N CHOOSE K VECTOR TO THE MAIN SUBROUTINE
C BINTST.
      REAL PSTAR,ALPHAT,NKV(21),PQV(21),PMF(21),CUMPR
      INTEGER N,T,HYP,I,TP1,NP1
      CALL NCHUZK(N,NKV)
      NP1=N+1
      CALL BINPMF(PSTAR,N,NKV,N,PQV,PMF,CUMPR)
      TP1=T+1
      ALPH1=0
      ALPH2=0
      IF (HYP.EQ.3) GO TO 20
         DO 15 I=TP1,NP1
               ALPH1=ALPH1+PMF(I)
 15      CONTINUE
         ALPH=ALPH1
 20   IF (HYP.EQ.2) GO TO 30
         DO 25 I=1,TP1
               ALPH2=ALPH2+PMF(I)
 25      CONTINUE
         ALPH=ALPH2
 30   IF(HYP.NE.1) GO TO 99
         IF(ALPH.GT.ALPH1) ALPH=ALPH1
         ALPH=ALPH*2.0
 99   WRITE(9,101) ALPH
 101  FORMAT(1X, ' VALUE OF ALPHA HAT:',F10.3)
      ALPHAT=ALPH
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE CIBINO(N,T,ALPHYP,NKV,CILL,CIUL)
C     THIS USES A BISECTION SEARCH TO FIND THE UPPER AND
C LOWER LIMITS (CILL,CIUL) TO THE PROBABILITY OF SUCCESS
C GIVEN THE NUMBER OF SUCCESSES T AND TOTAL SAMPLE SIZE N.
C NKV FROM NCHUZK.
C   THIS USES SYMMETRY OF BINOMIAL, E.G., A DIST WITH
C 8 SUCCESSES OF 10 WILL HAVE THE LIMITS OF PROBABILITIES
C 1.0 - LIMITS OF A DIST WITH 2 SUCC OF 10 (UPPER<=>LOWER)
      INTEGER N,T,TT,I,J,TTM1,LCOUNT
      REAL ALPHYP,NKV(21),CILL,CIUL,MLEP,LALPH,UALPH
      UALPH=ALPHYP*0.5
      LALPH=1.0-UALPH
      NDIV2=IFIX(FLOAT(N+1)*0.5)
      TT=T
      IF(T.GT.NDIV2) TT=N-T
      TTM1=TT-1
      MLEP=FLOAT(TT)/FLOAT(N)
      PLL=0.0
      PUL=0.0
      IF (TTM1.GE.0) PLL=BISEX(N,TTM1,NKV,LALPH)
      IF (TT.LE.N) PUL=BISEX(N,TT,NKV,UALPH)
      IF(T.LE.NDIV2) GO TO 10
         CILL=1.0-PLL
         CIUL=1.0-PLL
         GO TO 20
 10   CILL=PLL
```

```
      CIUL=PUL
      IF (CILL.LT.0.0) CILL=0.0
      IF (CIUL.GT.1.0) CIUL=1.0
 20   WRITE (9,101) CILL,CIUL
 101  FORMAT(1X,' PERCENT LOWER LIMIT',F10.3,
    * ' UPPER LIMIT',F10.3)
      RETURN
      END
C---------------------------------------------------------
      FUNCTION BISEX(N,T,NKV,ALPH)
C THIS DOES A RAY SEARCH TO FIND THE EXACT P FOR WHICH
C THERE IS A 1.0-ALPHA/2 OR ALPHA/2 PROB OF T SUCCESSES IN
C N TRIALS. NKV IS VECTOR OF N CHOOSE K,K=0,...,N
      REAL NKV(21),ALPH,LO,MED,HI,EPSILN,CUMPR,PQV(21),
     *GAP,P,EPSM1,PMF(21)
      INTEGER N,T,LCOUNT,LLIM
      EPSILN=0.002
      EPSM1=1.0-EPSILN
      LCOUNT=0
      LLIM=64
      IF(ALPH.LT.0.5) GO TO 10
         LO=0.0
         HI=FLOAT(T)/FLOAT(N)
         GO TO 20
 10   LO=FLOAT(T)/FLOAT(N)
      HI=1.0
 20   MED=0.5*(LO+HI)
      WRITE(9,201) LO,MED,HI,LCOUNT
      IF ((MED.LE.EPSILN).OR.(MED.GE.EPSM1)) GO TO 95
 201  FORMAT(1X,' LO,MED,HI, LOOP COUNT',3F10.5,I5)
      LCOUNT=LCOUNT+1
      CALL BINPMF(MED,N,NKV,T,PQV,PMF,CUMPR)
      GAP=CUMPR-ALPH
      IF((ABS(GAP).LT.EPSILN).OR.(LCOUNT.GT.LLIM)) GO TO 95
      IF(GAP.LT.0.0) GO TO 30
         LO=MED
         GO TO 20
 30   HI=MED
      GO TO 20
 95   WRITE (9,101) ALPH,MED
 101  FORMAT(1X,' ALPHA:',F10.3,' CI LIMIT:',F10.3)
      BISEX=MED
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/
      SUBROUTINE QUANTL(NOF,RCCT,RP,IP,ALPHAT,CILL,CIUL,
     *  LIST,LPTR,TIEV,TYPCEL)
C     THIS FINDS THE PROBABILITY THAT A SPECIFIED X,XSTAR
C IS THE SPECIFIED QUANTILE POINT QSTAR IN A DISTRIBUTION
C REPRESENTED BY THE VALS OF SAMPLE. THE CONFIDENCE
C INTERVALS (WHICH ENCLOSE POPULATION QSTAR) ARE GIVEN
      INTEGER NOF,RCCT(50),IP(50),LPTR(17),TIEV(2),T1,T2,
     *   T2P2,T1P1,N,NP1,RHAT,SHAT,KEY(400),TYPCEL,HYP
      REAL RP(50),ALPHAT,CILL,CIUL,LIST(400),NKV(21),PSTAR,
     * PMF(21),ALPH1,ALPH2,ALFMIN,ZHAT1,ZHAT2,PQV(21),
     * CUMPR,ZSTAR,NP,CONFCC,LTAIL,UTAIL,ZRHAT,ZSHAT,ALPHYP
      WRITE(NOF,523)
 523  FORMAT(10X,'**QUANTILE TEST**')
      T1=RCCT(1)-TIEV(1)
      T2=RCCT(1)+TIEV(2)
      HYP=IP(1)
      PSTAR=RP(1)
      ALPHYP=1.0-RP(2)
      HALFA=0.5*ALPHYP
      N=RCCT(2)+RCCT(1)
      WRITE(NOF,334) T1,T2,N
 334  FORMAT(1X,' NUMBER OF DATA ELEMENTS LESS THAN X*',I5/
```

```
     *1X,' NUMBER OF ELEMENTS LESS THAN OR EQUAL TO X*',
     *I5/1X,' TOTAL NUMBER OF ELEMENTS',I5)
      NP1=N+1
      IF (N.GT.20) GO TO 30
         CALL NCHUZK(N,NKV)
         CALL BINPMF(PSTAR,N,NKV,N,PQV,PMF,CUMPR)
         CALL CI20Q (ALPHYP,N,PMF,RHAT,SHAT,CONFCO)
         WRITE (9,102) T1,T2,RHAT,SHAT
102      FORMAT(1X,' T1,T2,RHAT,SHAT',4I5,' PMF:')
         WRITE(9,103) (PMF(I),I=1,NP1)
103      FORMAT(1X,10F6.3)
         T2P2=T2+2
         T1P1=T1+1
         ALPHA1=0.0
         ALPHA2=0.0
         DO 10 I=1,T1P1
              ALPHA1=ALPHA1+PMF(I)
10       CONTINUE
         DO 15 I=T2P2,NP1
              ALPHA2=ALPHA2+PMF(I)
15       CONTINUE
         WRITE(9,104) ALPHA1,ALPHA2
104      FORMAT(1X,' ALPHA1, ALPHA2:',2F7.3)
         GO TO 40
30    STDEV=(FLOAT(N)*PSTAR*(1.0-PSTAR))**0.5
         NP=FLOAT(N)*PSTAR
         ZHAT1=(T1-NP)/STDEV
         CALL NORMV(1,ZHAT1,ALPH1)
         ZHAT2=(T2-NP)/STDEV
         CALL NORMV(1,ZHAT2,ALPH2)
         ALPH2=1.0-ALPH2
         CALL NORMV(2,ZSTAR,HALFA)
         RHAT=1+(IFIX(NP+ZSTAR*STDEV-EPSILN))
         SHAT=1+(IFIX(NP-ZSTAR*STDEV-EPSILN))
         ZRHAT=(RHAT-NP)/STDEV
         CALL NORMV(1,ZRHAT,LTAIL)
         ZSHAT=(SHAT-NP)/STDEV
         CALL NORMV(1,ZSHAT,UTAIL)
         UTAIL=1.0-LTAIL
         CONFCO=1.0-LTAIL-UTAIL
         WRITE(9,108) CONFCO,LTAIL,UTAIL
108      FORMAT(1X,' CONFCO,LTAIL,UTAIL',3F7.3)
40    ALFMIN=ALPHA1
      IF (ALPHA1.GT.ALPHA2) ALFMIN=ALPHA2
         WRITE(9,174) ALPHA1,ALPHA2,ALFMIN
174      FORMAT(1X,' ALPHA1, ALPHA2, ALFMIN:',3F7.3)
      IF (HYP.EQ.1) ALPHAT=2.0*ALFMIN
      IF (HYP.EQ.2) ALPHAT=ALPH2
      IF (HYP.EQ.3) ALPHAT=ALPH1
      DO 5 I=1,N
           KEY(I)=I
5     CONTINUE
      IF (TYPCEL.EQ.5) GO TO 95
      CALL SHSORT (LIST,KEY,N)
      WRITE (9,103) (LIST(I),I=1,N)
      CILL=LIST(RHAT)
      IF(SHAT.GT.N) SHAT=N
      CIUL=LIST(SHAT)
      WRITE (9,101) ALPHAT,CILL,CIUL
      WRITE (NCF,101)ALPHAT,LIST(RHAT),LIST(SHAT),RHAT,SHAT
101   FORMAT(1X,'ALPHA HAT,LOWER AND UPPER BOUNDS',3F9.3/1X,
     *'INDEX OF LOWER LIMIT',I5,', INDEX OF UPPER LIMIT',I5)
      GO TO 99
95    WRITE (9,107) ALPHAT,RHAT,SHAT
      WRITE (NCF,107) ALPHAT,RHAT,SHAT
107   FORMAT(1X,'ALPHA HAT',F7.3,
     *' INDICES OF LOWER AND UPPER BOUNDS',2I5)
```

```
 99   WRITE (9,109) CONFCO
      WRITE (NCF,109) CCNFCC
 109  FORMAT(1X,'INTERVAL CONFIDENCE COEFFICIENT:',F10.3)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE CI2OQ (ALPHYP,N,PMF,RHAT,SHAT,CONFCO)
C   THIS SUBROLTINE FINDS INDICES RHAT AND SHAT THAT WILL
C INCLUDE WILL GIVE A CONFIDENCE COEFFICIENT OF AT LEAST
C 1.0-ALPHA, USED FCR QUANTILE TEST.
C RHAT IS INDEX CF LOWER LIMIT, SHAT IS INDEX OF UPPER
C BOUNC.  SINCE INDICES OF PMF ARE 1
C TO N+1, S=SHAT, R=RHAT
      INTEGER RHAT,SHAT,RMAX,SMIN,N,NP1,I,IM1,J,JP1
      REAL PMF(21),CDF(23),RCDF(23),ALPHYP,HALFA,BEST,CRIT,
     *  CRIT1,CRIT2,CONFCO
      NP1=N+1
      HALFA=0.5*ALPHYP
      I=1
      CDF(1)=PMF(1)
 10   CONTINUE
          IM1=I
          IF (CDF(I).GE.HALFA) GO TO 15
          I=I+1
          CDF(I)=CDF(IM1)+PMF(I)
          WRITE(9,191) I,PMF(I),CDF(I)
 191      FCRMAT(1X,'I,PMF(IM1),CDF(I)',I5,2F10.4)
      IF (CDF(I).LE.HALFA) GO TO 10
 15   RHAT=IM1
      J=NP1+1
      RCDF(J)=0
 20   CONTINUE
          JP1=J
          J=J-1
          RCDF(J)=RCDF(JF1)+PMF(J)
          WRITE(9,192) J,RCDF(J),PMF(J)
 192      FCRMAT(1X,'J,RCDF(J),PMF(J)',I5,2F10.4)
      IF (RCDF(J).LE.HALFA) GO TO 20
      SHAT=J
      CONFCO=1.0-CDF(RHAT)-RCDF(SHAT)
      WRITE (9,101) CDF(RHAT),RCDF(SHAT),CCNFCC,RHAT,SHAT
 101  FORMAT(1X,'LO,UP,CONFCO,INDICES, CI:',/1X,
     * 3F10.4,2I5)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE COXSTU (LIST,LPTR,IP,RP,NCF,NTEST)
C     THIS SUBRCUTINE DIVIDES THE ONE VARIABLE IN TWO, THEN
C PERFORMS THE SIGN TEST TC TEST THE TREND AS A MATTER OF
C PAIRING THE FIRST HALF DATA ELEMENTS WITH THE SECCND HALF
C THERE ARE TWO CPTIONS: 1=ONE VARIABLE, IN PRESENT ORDER,
C 2=CORRELATICN, RANKS OF THE INDEPENDENT VARIABLE
C ORDERING THE DEPENDENT VARIABLE
      INTEGER LPTR(17),IP(50),NOF,TLPTR(3),NHALF,MED,NHT2,
     *    HYP,INDEPC,DEPC,KEY(200),NACTUL
      REAL LIST(400),RP(50),TLIST(400),SLIST(400),DEP(200),
     * INDEP(200)
      WRITE(NOF,523)
      WRITE(9,523)
 523  FORMAT(10X,'**COX AND STUART TEST FOR TREND**')
      IF (IP(2).EQ.1) GO TO 7
      N=LPTR(2)-1
      INDEPC=0
      DEPC=0
      IF(IP(3).LT.IP(4)) INDEPC=N
      IF(IP(4).LT.IP(3)) DEPC=N
      DO 5 I=1,N
```

```
            KEY(I)=I
            INDEP(I)=LIST(I+INDEPC)
            DEP(I)=LIST(I+DEPC)
 5       CONTINUE
         CALL SHSORT(INDEP,KEY,N)
         DO 6 I=1,N
            SLIST(I)=DEP(KEY(I))
            WRITE(9,918) I,KEY(I),SLIST(I)
 918        FORMAT(1X,'IN COXSTU:I,KEY,SLIST:',2I5,F10.3)
 6       CONTINUE
         GO TO 9
 7       DO 8 I=1,N
            SLIST(I)=LIST(I)
 8       CONTINUE
 9       NHALF=IFIX(0.5*(LPTR(2)-1))
         NHT2=2*NHALF
         TLPTR(1)=1
         TLPTR(2)=1+NHALF
         TLPTR(3)=1+NHT2
         NACTUL=LPTR(2)-1
         HYP=IP(1)
         DO 10 I=1,NHALF
            TLIST(I)=SLIST(I)
 10      CONTINUE
         IF (NACTUL.GT.NHT2) GO TO 30
         DO 20 I=NHALF,NACTUL
            TLIST(I)=SLIST(I)
 20      CONTINUE
         GO TO 50
 30      LL=2+NHALF
         DO 40 I=LL,NACTUL
            IM1=I-1
            TLIST(IM1)=SLIST(I)
 40      CONTINUE
         IF(HYP.EQ.1) WRITE (NOF,111)
 111     FORMAT(1X,' NULL HYPOTHESIS: THERE IS NO TREND ')
         IF (HYP.EQ.2) WRITE (NOF,112)
 112     FORMAT(1X,'NULL HYPOTHESIS: THERE IS NO UPWARD TREND')
         IF (HYP.EQ.3) WRITE (NOF,113)
 113     FORMAT(1X,
     *     'NULL HYPOTHESIS: THERE IS NO DOWNWARD TREND')
 50      CALL SGNT(TLIST,TLPTR,NOF,NHT2,RP,IP,NTEST)
         RETURN
         END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/
         SUBROUTINE SGNT(LIST,LPTR,NOF,TOTELM,RP,IP,NTEST)
         REAL A(200),B(200),RP(50),LIST(400),P,EPSILN,ADIF,DIF,
     *NKV(21),ALPHAT,ALPHYP,CIUL,CILL,ALPHA1,ALPHA2
         INTEGER LPTR(17),I,J,NELMV,IE,K,M,NPLUS,NMINUS,NTOT,
     *HYP,IP(50),NTEST,NOF
C        NELMV IS THE NUMBER OF ELEMENTS IN VECTORS A AND B
         NPLUS=0
         NMINUS=0
         IF (NTEST.EQ.5) WRITE(NOF,523)
 523     FORMAT(10X,'**SIGN TEST**')
         NELMV=LPTR(2)-1
         EPSILN=0.0001
         HYP=IP(1)
         IF (HYP.EQ.1) WRITE (NOF,111)
         IF (HYP.EQ.1) WRITE (9,111)
 111     FORMAT(1X,' H0: P(+)=P(-)'/1X,' H1: P(+)<>P(-)')
         IF (HYP.EQ.2) WRITE (NOF,112)
 112     FORMAT(1X,' H0: P(+)<=P(-)'/1X,' H1: P(+)>P(-)')
         IF (HYP.EQ.3) WRITE (NOF,113)
 113     FORMAT(1X,' H0: P(+)>=P(-)'/1X,' H1: P(+)<P(-)')
         ALPHYP=1.0-RP(1)
         DO 14 I=1,NELMV
```

```
            J=LPTR(2)-1+I
            DIF=LIST(I)-LIST(J)
            WRITE(9,917) I,J,LIST(I),LIST(J),DIF
  917       FORMAT(1X,'IN SIGN:I,J,L(I-J),DIF:',2I5,3F10.3)
            ADIF=ABS(DIF)
            IF(ADIF.LE.EPSILN) GO TO 14
            IF (DIF.LT.0) NPLUS=NPLUS+1
            IF (DIF.GT.0) NMINUS=NMINUS+1
  14      CONTINUE
          WRITE(9,987) NPLUS,NMINUS
          WRITE(NOF,986) NPLUS,NMINUS
  987     FORMAT(1X,'SIGN TEST +,-',2I5)
  986     FORMAT(1X,' NUMBER OF (+):',I5,' NUMBER OF (-):',I5)
          NTOT=NPLUS+NMINUS
          IF (NTOT.LE.20) CALL NCHUZK(NTOT,NKV)
          IF (NTOT.LE.20) CALL BINH20
         *            (NTOT,NPLUS,0.5,HYP,ALPHAT,NKV)
          IF (NTOT.GT.20) CALL
         *BINORM(NTOT,NPLUS,0.5,HYP,ALPHAT,ALPHYP,CILL,CIUL)
C         ALPHA2=1.0-ALPHA1
C         ALFMIN=ALPHA1
C         IF (ALPHA1.GT.ALPHA2) ALFMIN=ALPHA2
C         IF (HYP.EQ.1) ALPHAT=2.0*ALFMIN
C         IF (HYP.EQ.2) ALPHAT=ALPH2
C         IF (HYP.EQ.3) ALPHAT=ALPH1
          WRITE(NOF,102) NELMV,NTOT,NPLUS,ALPHAT,ALPHYP
  102     FORMAT(1X,' NUMBER OF PAIRS:',I5,' N OF ',
         *'PAIRED VALUES NOT TIED',I5,/' NUMBER OF PLUSSES',I5,
         */' ALPHA HAT:',F10.4/' HYPOTHESISED ALPHA ',F10.4)
          IF(ALPHAT.LT.ALPHYP) WRITE(NOF,104)
          IF(ALPHAT.GE.ALPHYP) WRITE(NOF,105)
  104     FORMAT(1X,' REJECT THE NULL HYPOTHESIS ')
  105     FORMAT(1X,' DO NOT REJECT THE NULL HYPOTHESIS')
  99      RETURN
          END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/
          SUBROUTINE MANWHT(LPTR,LIST,NOF,RP)
          INTEGER N1,N2,NT,NHOLD,I,J,LPTR(17),IER
          REAL A(400),LIST(400),RP(1),U,Z,R(400),ALPH,ALPHAT
          WRITE(NOF,523)
  523     FORMAT(10X,'**MANN WHITNEY TEST**')
          N1=LPTR(2)-LPTR(1)
          N2=LPTR(3)-LPTR(2)
          NT=N1+N2
          IF(N1.LE.N2)GO TO 30
            DO 10 I=1,NT
                J=LPTR(3)-I
                A(I)=LIST(J)
  10        CONTINUE
            NHOLD=N2
            N2=N1
            N1=NHOLD
            GO TO 35
  30      DO 32 I=1,NT
            A(I)=LIST(I)
  32      CONTINUE
  35      CALL UTEST (A,R,N1,N2,U,Z,IER)
          IF(IER.LT.1) GO TO 40
           WRITE (NOF,103)
  103      FORMAT (1X,' ALL VALUES OF ONE SUBGROUP ARE TIED')
  40       IF (N2.LT.20) GO TO 95
           CALL NORMV(1,Z,ALPHAT)
          WRITE (NOF,101) U,Z,ALPHAT
  101     FORMAT(1X,' TEST STATISTIC U:',F10.4,
         *  'STANDARDIZED Z  :',F10.4/1X,
         *            ' ALPHA HAT:',F10.4)
          ALPH=1.0-RP(1)
```

```
      IF(ALPHAT.LT.ALPH) WRITE(NOF,104)
      IF(ALPHAT.GE.ALPH) WRITE(NOF,105)
 104  FORMAT(1X,' REJECT THE NULL HYPOTHESIS ')
 105  FORMAT(1X,' DO NOT REJECT THE NULL HYPOTHESIS')
      GO TO 99
 95   WRITE(401) U,N1,N2
 401  FORMAT(1X,' TEST STATISTIC U:',F10.3/1X,
     *' WITH N1=',I4,', AND N2=',I4)
 99   RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE MCNEMR(RCCT,RP,IP,NOF)
C     THIS SUBROUTINE FIGURES THE VALUES OF THE MCNEMAR TEST:
C IF N<21, IT USES THE EXACT VALUES OF THE BINOMIAL,ELSE
C USES THE LINEAR INTERPOLATION OF CHI-SQUARE DF=1
      INTEGER RCCT(4),IP(50),B,C,N,T2,NKV(21)
      REAL RP(50),ALPHAT,ALPHYP,T1
      WRITE (NOF,523)
 523  FORMAT(10X,'**MCNEMAR TEST**')
      B=RCCT(2)
      C=RCCT(3)
      ALPHYP=1.0-RP(1)
      T2=B
      N=B+C
      IF (N.GT.20) GO TO 30
         CALL BINH20(N,B,0.5,1,ALPHAT,NKV)
         ALPHAT=0.5*ALPHAT
         WRITE (NOF,103) B,ALPHYP,ALPHAT
 103     FORMAT(1X,' T2 ',I5,
     *      ' HYPOTHESISED ALPHA,ALPHA HAT ',2F7.3)
         GO TO 40
 30   T1=(FLOAT((B-C)**2))/FLOAT(B+C)
      CALL CHIONE (T1,ALPHAT)
         WRITE (NOF,104) T1,ALPHYP,ALPHAT
 104     FORMAT(1X,' T1 ',F9.3,
     *      ' HYPOTHESISED ALPHA,ALPHA HAT ',2F7.3)
 40   IF (ALPHAT.LE.ALPHYP) WRITE (NOF,101)
 101  FORMAT(1X,' REJECT THE NULL HYPOTHESIS')
      IF (ALPHAT.GT.ALPHYP) WRITE (NOF,102)
 102  FORMAT(1X,' DO NOT REJECT THE NULL HYPOTHESIS')
      WRITE (NOF,111) (RCCT(I),I=1,4)
 111  FORMAT(1X,' MCNEMAR TABLE:'/25X,'AFTER:'/1X,'BEFORE:',
     *12X,'SUCCESS    FAILURE'/1X,'SUCCESS',3X,2I10/1X,
     *'FAILURE',3X,2I10)
      RETURN
      END
C----------------------------------------------------------
      SUBROUTINE CHIONE (X,ALPHAT)
C        THIS SUBROUTINE FINDS AN APPROXIMATION FOR
C CHI-SQUARE DF=1 BY LINEAR INTERPOLATION
      REAL CHI(8),ALPHAT,P(8)
      INTEGER N
      CHI(1)=0
      CHI(2)=1.323
      CHI(3)=2.706
      CHI(4)=3.841
      CHI(5)=5.024
      CHI(6)=6.635
      CHI(7)=7.879
      CHI(8)=10.83
      P(1)=0
      P(2)=0.75
      P(3)=0.9
      P(4)=0.95
      P(5)=0.975
      P(6)=0.99
      P(7)=0.995
```

```
      P(8)=0.999
 10   CONTINUE
         I=I+1
      IF ((X.GT.CHI(I)).AND.(I.LT.8)) GO TO 10
      IF (I.LT.8) GO TO 20
         PHAT=0.999
         GO TO 95
 20   IM1=I-1
      RATIO=(X-CHI(IM1))/(CHI(I)-CHI(IM1))
      PHAT=P(IM1)+RATIO*(P(I)-P(IM1))
 95   ALPHAT=1.0-PHAT
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/
      SUBROUTINE FRIEDM (NTREAT,LPTR,LIST,IP,RP,NOF,TOTELM)
C   THIS DOES EITHER THE FRIEDMAN AND DURBIN TESTS;
C IP(1) =0, FRIEDMAN, =1, DURBIN; IF DURBIN, RP(2)=FILLER,
C IP(2)=K=# TREATS PER BLOCK; IP(3)=R=# BLOCKS PER TREAT
C FOR BOTH, RP(1)=CONF COEF.  RANQUE RANKS THE VALUES
C BLOCK BY BLOCK; PARCOM COMPARES THE SUMS OF RANKS OF
C DIFFERENT TREATMENTS TO TEST FOR PAIRWISE DIFFERENCE
      INTEGER NOF,NTREAT,LPTR(17),K,B,I,J,RI,TOTELM,
     *DF1,DF2,KEY(400),NBLOCK,INTEST(400),T,R,IP(50)
      REAL LIST(400),T2,SVALS(400),RK(400),B2,A2,RJ,
     *EPSILN,BK25,BM1,DENOM,NUM,MP1,SRJS,NUMVAR,DENVAR,
     *STDEV,SRJ(17),TERM1,TERM2,RP(50),Q(17)
      DO 3 I=1,TOTELM
         INTEST(I)=1
 3    CONTINUE
      MP1=-999999.9
      IF (IP(1).EQ.1) GO TO 5
      WRITE(NOF,524)
 524  FORMAT(10X,'**FRIEDMAN TEST**')
      GO TO 8
 5    WRITE(NOF,625)
 625  FORMAT(10X,'**DURBIN TEST**')
      DO 6 I=1,TOTELM
         IF (LIST(I).GT.MP1) MP1=LIST(I)
 6    CONTINUE
      MP1=MP1+1.0
      R=IP(3)
      K=IP(2)
      DO 7 I=1,TOTELM
         IF (LIST(I).NE.RP(2)) GO TO 7
            LIST(I)=MP1
            INTEST(I)=0
 7    CONTINUE
      WRITE(9,919) RP(2),MP1
 919  FORMAT(1X,' INTEST VALUES, DURBIN; RP(2),MP1',2F10.3)
      WRITE(9,818) (INTEST(I),I=1,TOTELM)
 818  FORMAT(1X,12I5)
 8    EPSILN=0.001
      NBLOCK=LPTR(2)-1
      CALL RANQUE
     *  (TOTELM,LIST,SVALS,RK,Q,KEY,NTREAT,NBLOCK,2)
      WRITE(9,315) (RK(I),I=1,TOTELM)
 315  FORMAT(1X,6F10.3)
      SRJS=0.0
      A2=0.0
      B=LPTR(2)-1
      T=NTREAT
      DO 10 J=1,T
         RJ=0
         DO 15 I=1,B
            RI=I+LPTR(J)-1
            IF(INTEST(RI).EQ.0) GO TO 15
            RJ=RJ+RK(RI)
```

```
                A2=A2+(RK(RI)**2)
                WRITE (9,312) I,J,RI,RJ,A2
 312            FCRMAT(1X,'IJ,RI',3I5,'RJ,A2',2F10.3)
 15         CONTINLE
            SRJ(J)=RJ
            SRJS=SRJS+(RJ*RJ)
 10     CONTINUE
        IF (IP(1).EQ.1) GC TO 50
        K=T
        B2=SRJS/(FLOAT(B))
        DENOM=A2-B2
        WRITE(9,316) A2,B2,DENOM
        WRITE(NOF,316) A2,B2,DENOM
 316    FORMAT(1X,'A2,B2,DENOM',3F10.3)
        IF (DENOM.LE.EPSILN) GO TO 99
        BM1=FLOAT(B-1)
        BK25=0.25*(FLCAT(B*K*((K+1)**2)))
        NUM=BM1*(B2-BK25)
        T2=NUM/DENCM
        DF1=K-1
        CF2=(B-1)*(K-1)
        WRITE (NCF,112) T2,DF1,DF2
        WRITE (9,112) T2,CF1,DF2
 112    FORMAT(1X,'VALUE OF T2:',F10.3,'  DF1:',I5,' DF2:',I5)
        NUMVAR=2.C*B*(A1-B1)
        DENVAR=(K-1)*(B-1)
        STDEV=(NUMVAR/DENVAR)**0.5
        CALL PARCCM(SRJ,T,STDEV,NCF)
        GO TO 99
 50     CONTINUE
        DF1=12*(T-1)
        CF2=(R*T*(K-1)*(K+1))
        TERM1=SRJS*FLCAT(DF1)/FLCAT(DF2)
        DF1=T-1
        CF2=3*R*CF1*(K+1)/(K-1)
        TERM2=FLCAT(DF2)
        T3=TERM1-TERM2
        WRITE(9,101) TERM1,TERM2,SRJS,T3
 101    FORMAT(1X,'TERM1-2,SRJS,T3',4F10.4)
        WRITE(NOF,102) T3,DF1
 102    FORMAT(1X,' TEST STATISTIC T3',F10.4,/1X,
      *' TO BE CCMPARED WITH CHISQUARED DISTRIBUTION'/1X,
      *' WITH DEGREES OF FREEDOM=',I5)
        CF2=B*K*CF1
        NUMVAR=FLCAT((R*(K+1)*(K-1)))
        NUMVAR=NUMVAR*(FLCAT(CF2)-T3*FLOAT(T))
        CF2=6*DF1*(6*K-T-B+1)
        CENVAR=FLCAT(CF2)
        STDEV=(NUMVAR/DENVAR)**0.5
        CALL PARCCM(SRJ,T,STDEV,NOF)
 99     CONTINUE
        RETURN
        END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
        SUBROUTINE QUADE (NTREAT,LPTR,LIST,NCF,TCTELM)
C  THIS PERFORMS THE QUADE TEST.  RANQUE RANKS THE DATA
C ELEMENTS BLOCK BY BLOCK AND RANKS THE RANGES OF THE
C BLOCK ELEMENTS (INTO Q).  PARCOM HAS PAIRWISE COMPARISON
C OF SUMS OF TREATMENT RANKS, TO TEST FOR EQUAL EFFECT
        INTEGER NCF,NTREAT,LPTR(17),K,B,I,J,RI,K,B,TCTELM,
      *  DF1,DF2,KEY(400),NBLOCK
        REAL LIST(TOTELM),T2,SVALS(400),R(400),B2,A2,RJ,NUM,
      * EPSILN,BK25,BM1,Q(200),A1,B1,S(400),SIJ,SJ2,SJ,MEAN,
      * DENCM,DENVAR,NUMVAR,STDEV,SRJ(17)
        WRITE(NOF,524)
 524    FORMAT(10X,'**QUADE TEST**')
        K=NTREAT
```

```
      B=LPTR(2)-1
      A1=0
      B1=0
      SJ2=0
      T1=0
      DF1=K-1
      DF2=DF1*(B-1)
      EPSILN=0.001
      MEAN=0.5*(FLOAT((K+1)))
      CALL RANQUE(TOTELM,LIST,SVALS,R,Q,KEY,K,B,3)
      DO 10 J=1,K
         SJ=0
         DO 15  I=1,B
             RI=I+LPTR(J)-1
             SIJ=Q(I)*(R(RI)-MEAN)
             A1=A1+SIJ*SIJ
             SJ=SJ+SIJ
            WRITE(9,101) I,J,RI,R(RI),Q(I),SIJ,A1
 101        FORMAT(1X,' I,J,RI',3I3,' R,Q(I),SIJ,A1',4F9.3)
 15      CONTINUE
        SRJ(J)=SJ
        SJ2=SJ2+SJ*SJ
 10   CONTINUE
      B1=SJ2/(FLOAT(B))
      DIF=A1-B1
      IF (DIF.LT.EPSILN) GO TO 95
      T1=(FLOAT(B-1))*B1/DIF
 95   WRITE (9,102) A1,B1,T1,DF1,DF2
      WRITE (NOF,102) A1,B1,T1,DF1,DF2
 102  FORMAT(1X,'A1',F10.3,' B1',F10.3/
     * 1X,' TEST STATISTIC T1:',F10.3,' DF1',I5,' DF2',I5)
      NUMVAR=2.0*(A1-B1)*FLOAT(B)
      DENVAR=FLOAT(DF2)
      STDEV=(NUMVAR/DENVAR)**0.5
      CALL PARCOM(SRJ,K,STDEV,NOF)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE COCHRN(NROW,NCOL,RCCT,IP,RP,NOF)
C        THIS SUBROUTINE TAKES ROWAND COLUMN COUNTS RCCT
C     FOR SUCCESSES, DERIVED IN CELLER, AND PERFORMS THE
C     COMPUTATION OF THE CHI SQUARED STATISTIC
      INTEGER SCJSQ,SRISQ,RCCT(50),NSUCC,NROW,NCOL,NUM,
     *NOF,IP(50),DENOM,DF
      REAL CHI,RP(50)
      NSUCC=0
      SCJSQ=0
      SRISQ=0
      DO 10 J=1,NCOL
         IC=J+NROW
         SCJSQ=SCJSQ+RCCT(IC)**2
 10   CONTINUE
      DO 20 I=1,NROW
         NSUCC=NSUCC+RCCT(I)
         SRISQ=SRISQ+RCCT(I)**2
 20   CONTINUE
      WRITE(NOF,101) SRISQ,SCJSQ,NSUCC,NCOL
 101  FORMAT(10X,'**COCHARAN TEST**',/1X,
     *'SUM OF THE SQUARES: R(I)',I5,', C(J)',I5/1X,'FOR',I5,
     * ' SUCCESSES IN',I5,' TREATMENTS')
      NUM=(NCOL*(NCOL-1)*SCJSQ)-((NCOL-1)*NSUCC*NSUCC)
      DENOM=(NCOL*NSUCC)-SRISQ
      CHI=FLOAT(NUM)/FLOAT(DENOM)
      DF=NCOL-1
      WRITE(NOF,102) NUM,DENOM,CHI,DF
 102  FORMAT(1X,' NUMERATOR,DENOMINATOR:',2I5/1X,' CHI ',
     *'SQUARED VALUE:',F10.3/1X'DEGREES OF FREEDOM:',I5)
```

```
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE RCCNT(RCCT,NROW,NCOL,NOF)
C   THIS SUBROUTINE PERFORMS THE CHI-SQUATED TEST ON CELL
C COUNTS GIVEN BY THE VECTOR RCCT (ROW & COLUMN CELL COUNT)
      INTEGER RCCT(50),NROW,NCOL,NOF,MAX,RC(50),CC(50),DF,
     *UL,LL
      REAL EIJ,CHI
      MAX=NROW
      IF(NCOL.GT.NROW) MAX=NCOL
      DO 10 I=1,MAX
         RC(I)=0
         CC(I)=0
 10   CONTINUE
      WRITE(9,102) NROW,NCOL
      WRITE(NOF,102) NROW,NCOL
 102  FORMAT(1X,' IN RCCNT: NROW,NCOL:',2I5,' RCCT:')
      DO 15 I=1,NROW
         LL=1+NCOL*(I-1)
         UL=NCOL*I
         WRITE(9,103) (RCCT(K),K=LL,UL)
         WRITE(NOF,103) (RCCT(K),K=LL,UL)
 103     FORMAT(1X,16I5)
 15   CONTINUE
      N=0
      DO 20 I=1,NROW
         DO 22 J=1,NCOL
               K=J+(I-1)*NCOL
               CC(J)=CC(J)+RCCT(K)
               RC(I)=RC(I)+RCCT(K)
               N=N+RCCT(K)
 22      CONTINUE
 20   CONTINUE
      CHI=0.0
      DO 30 I=1,NROW
         DO 32 J=1,NCOL
               EIJ=(FLOAT(CC(J)*RC(I)))/FLOAT(N)
               K=J+(I-1)*NCOL
               CHI=CHI+((FLOAT(RCCT(K))-EIJ)**2)/EIJ
         WRITE(9,678) I,J,K,EIJ,RCCT(K)
 678     FORMAT(1X,'I,J,K,EIJ,CIJ:',3I5,2F9.3)
 32      CONTINUE
 30   CONTINUE
      DF=(NROW-1)*(NCOL-1)
      WRITE(9,101) CHI,DF
      WRITE(NOF,101) CHI,DF
 101  FORMAT(1X,' CHI-SQUARED VALUE',F10.3,
     * '  DEGREES OF FREEDOM',I5)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE GDFIT(LIST,LPTR,IP,RP,NOF)
C   THIS GROUP OF SUBS FINDS PARAMETERS IF NECESSARY AND
C COMPARES AN ACTUAL DISTRIBUTION TO A THEORETICAL DIST.
C AVAILABLE THEORETICAL DISTRIBUTIONS ARE NORMAL,UNIFORM,
C EXPONENTIAL.  SAMPLE SIZE 15 OR MORE DESIRED.
C IP(1)=# DIST; IP(2)=# ESTIMATED PARAMS;IP(3)=N OF ERLANG
C RP(1)=MU,A,LAMBDA; RP(2)=VAR,B,ALPHA; RP(3)=CONF COEF
      REAL RP(50),LIST(400),CUMP(80),PARTV(80),CHI,MU,VAR
      INTEGER LPTR(17),N,EJ(80),ESTPAR,NDIST,NCELL,NOF,DF,
     *IP(50),RCCT(80)
      WRITE(NOF,523)
 523  FORMAT(1X,'**GOODNESS OF FIT TEST (CHI SQUARE)**')
      N=LPTR(2)-1
      CHI=0.0
      CALL CPVGEN(N,CUMP,EJ,NCELL)
```

```
      CALL MUVAR(LIST,N,MU,VAR)
      RP(4)=MU
      RP(5)=VAR
      NDIST=IP(1)
      IF (NDIST.EQ.1)
     *   CALL NORMD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
      IF (NDIST.EQ.2)
     *   CALL UNID(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
      IF (NDIST.EQ.3)
     * CALL EXPD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
      IF (NDIST.EQ.4)
     *   CALL ERLD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
      IF (NDIST.EQ.5)
     *   CALL WEID(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
      CALL CELCNT(ROCT,LIST,N,NCELL,PARTV)
      WRITE(NOF,121)
 121  FORMAT(1X,' CELL NR,  EJ(J) ,  O(J)  , PARTITION VALUE
     *   27('-'))
      DO 25 I=1,NCELL
         WRITE(NOF,122) I,EJ(I),ROCT(I)
 122     FORMAT(1X,3(I5,'   |'))
         IF(I.LT.NCELL) WRITE (NOF,123) PARTV(I)
 123     FORMAT(1X,3('-------|'),F10.3)
 25   CONTINUE
         WRITE(NOF,124) N,N
 124     FORMAT(27('-')/9X,2I8)
      DO 10 I=1,NCELL
         CHI=CHI+FLOAT((EJ(I)-ROCT(I))**2)/FLOAT(EJ(I))
 10   CONTINUE
      DF=NCELL-IP(2)-1
      WRITE(9,101) N,NCELL,CHI,DF
      WRITE(NOF,101) N,NCELL,CHI,DF
 101  FORMAT(1X,'WITH SAMPLE LENGTH',I4,' DIVIDED INTO',I3,
     *' CELLS'/1X,'CHI-SQUARED VALUE OF',F10.3/1X,'DF=',I5)
      IF(IP(2).EQ.0) WRITE(NOF,112) RP(4),RP(5)
 112  FORMAT(1X/1X,' SAMPLE MEAN AND VARIANCE:',2F10.4)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE CPVGEN(N,CUMP,EJ,NCELL)
C THIS SUBROUTINE FINDS THE EXPECTED CELL COUNT VECTOR EJ
C AND THE THE CUMULATIVE PROBABILITY VECTOR CUMP
      REAL LIST(400),CUMP(80),PARTV(80),ND5,FN
      INTEGER LPTR(17),N,EJ(80),ESTPAR,NDIST,NCELL,NOF,IND5,
     *  CUM(80),INCEL,MER,REM
      FN=FLOAT(N)
      IF (N.GT.52) GO TO 23
      ND5=FN/5.0
      IND5=IFIX(ND5)
      REM=N-(5*IND5)
      NCELL=IND5
      IF (REM.GT.2) NCELL=IND5+1
      DO 12 I=1,NCELL
         EJ(I)=5
 12   CONTINUE
      IF (REM.EQ.0) GO TO 20
         IF(REM.LE.2) EJ(1)=6
         IF(REM.GE.3) EJ(1)=4
         IF(REM.EQ.2) EJ(NCELL)=6
         IF(REM.EQ.3) EJ(NCELL)=4
 20   CONTINUE
      GO TO 29
 23   NCELL=10
      INCEL=IFIX(FN/10.0)
      REM=N-10*INCEL
      MER=10-REM
      DO 24 I=1,10
```

```
         IF (I.LE.MER) EJ(I)=INCEL
         IF (I.GT.MER) EJ(I)=INCEL+1
 24   CONTINUE
 29   CUM(1)=EJ(1)
      CUMP(1)=CUM(1)/FN
      DO 30 I=2,NCELL
         IM1=I-1
         CUM(I)=EJ(I)+CUM(IM1)
         CUMP(I)=FLOAT(CUM(I))/FN
 30   CONTINUE
      WRITE (9,101) N,NCELL
 101  FORMAT(1X,' CUMP GENERATOR FOR N=',I5,
     * ' #CELLS',I5,' CUMP VECTOR:')
      WRITE(9,102) (CUMP(I),I=1,NCELL)
 102  FORMAT(1X,8F7.3)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE NORMD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C THIS SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PROB VECTOR
C CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
      REAL LIST(400),CUMP(80),PARTV(80),RP(50),STDEV,MU,VAR
      INTEGER N,NCELL,IP(50)
      WRITE(NOF,524) IP(2)
 524  FORMAT(1X,'NORMAL DIST',I3,' PARAMETERS ESTIMATED')
      IF (IP(2).EQ.0) GO TO 5
         MU=RP(4)
         VAR=RP(5)
         GO TO 10
 5    MU=RP(1)
      VAR=RP(2)
 10   STDEV=VAR**0.5
      NCM1=NCELL-1
      WRITE(9,111) MU,VAR,STDEV,NCM1
 111  FORMAT(1X,'MU,VAR,STDEV,NCM1',3F8.3,I5)
      DO 20 I=1,NCM1
         CALL NORMV(2,Z,CUMP(I))
         PARTV(I)=STDEV*Z+MU
         WRITE(9,121) I,CUMP(I),Z,PARTV(I)
 121     FORMAT(1X,'I,CUMP(I),Z,PARTV(I)',I5,3F8.3)
 20   CONTINUE
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE CELCNT(RCCT,LIST,N,NCELL,PARTV)
C THIS ROUTINE SORTS THE SAMPLE AND FIGURES THE CELL COUNT
C OF THE SAMPLE CORRESPONDING TO THE EJ'S OF THE THEORETICAL
C DISTRIBUTION
      REAL LIST(400),PARTV(80)
      INTEGER RCCT(80),N,NCELL,KEY(400)
      J=1
      DO 5 I=1,N
         KEY(I)=I
 5    CONTINUE
      CALL SHSORT(LIST,KEY,N)
      DO 6 I=1,NCELL
         RCCT(I)=0
 6    CONTINUE
      NCM1=NCELL-1
      DO 20 I=1,N
 8       IF(LIST(I).LE.PARTV(NCM1)) GO TO 10
               RCCT(NCELL)=RCCT(NCELL)+1
               GO TO 20
 10      CONTINUE
               IF(LIST(I).LE.PARTV(J)) GO TO 15
                  J=J+1
```

```
                 IF(J.LT.NCM1) GO TO 10
                 IF(J.EQ.NCM1) GO TO 8
 15         RCCT(J)=RCCT(J)+1
 20     CONTINUE
        RETURN
        END
C-----------------------------------------------------------------
        SUBROUTINE UNID(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C    THIS SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C  THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PROB VECTOR
C  CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
        REAL LIST(400),CUMP(80),PARTV(80),RP(50),STDEV,A,B,C
        INTEGER N,NCELL,IP(50)
        WRITE(NOF,524) IP(2)
 524    FORMAT(1X,'UNIFORM DIST,',I3,' PARAMETERS ESTIMATED')
        A=RP(1)
        B=RP(2)
        C=B-A
        NCM1=NCELL-1
        DO 20 I=1,NCM1
           PARTV(I)=CUMP(I)*C+A
 20     CONTINUE
        RETURN
        END
C-----------------------------------------------------------------
        SUBROUTINE EXPD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C   THIS SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C  THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PROB
C  VECTOR CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
        REAL LIST(400),CUMP(80),PARTV(80),RP(50),LAMBDA
        INTEGER N,NCELL,IP(50)
        WRITE(9,524) IP(2)
 524    FORMAT(1X,'EXPONENTIAL DISTRIBUTION,',I3,
     *' PARAMETERS ESTIMATED')
        LAMBDA=RP(1)
        IF (IP(2).EQ.0)  LAMBDA=1.0/RP(4)
        NCM1=NCELL-1
        DO 20 I=1,NCM1
           PARTV(I)=(ALOG(1-CUMP(I)))/(-LAMBDA)
 20     CONTINUE
        RETURN
        END
C-----------------------------------------------------------------
        SUBROUTINE ERLD(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C       THIS FINDS THE X TO CORRESPONED TO THE F OF CUMP
C  BY NEWTON-RAPHSON SEARCH IN ERLNUT, TO YIELD PARTV
        REAL RP(50),LIST(400),CUMP(80),PARTV(80),CHI,DN,F,X,
     *LAMBDA,FN
        INTEGER LPTR(17),N,EJ(80),ESTPAR,NDIST,NCELL,NOF,DF,
     *IP(50),RCCT(80),FACT,NCM1
        NCM1=NCELL-1
        FN=FLOAT(IP(3))
        LAMBDA=RP(1)
        IF(IP(1).EQ.1) LAMBDA=FN/RP(4)
        NN=IP(3)-1
        FACT=1
        DO 5 I=1,NN
           FACT=FACT*I
 5      CONTINUE
        FN=FLOAT(FACT)
        DO 10 I=1,NCM1
           F=CUMP(I)
           CALL ERLNUT(F,X,NN,FN,LAMBDA)
           PARTV(I)=X
 10     CONTINUE
        RETURN
        END
```

```
C-----------------------------------------------------------------
      SUBROUTINE ERLNUT(F,X,NN,FN,LAMBDA)
C   THIS IS THE NEWTON-RAPHSON TECH WHICH TAKES AN F AND
C RETURNS AN X; NN=N-1; FN=(N-1)
      REAL F,X,FN,LAMBDA,XLO,XHI,XMED,NF,DIF,ADIF,SLOPE,
     *L,LX,LXN,FACT
      INTEGER NN,N
      NF=FLOAT(NN)
      EPSILN=0.0001
      L=LAMBDA
      X=(NF+1.0)/L
      IF(F.GT.0.7) X=X*2*F
      IF(F.LT.0.3) X=X*3*F
      XLO=0
      XHI=100.0*X
 10   CONTINUE
         LX=L*X
         LXN=0
         FACT=1
         DO 15 J=1,NN
               FACT=FACT*FLOAT(J)
               LXN=LXN+(LX**J)/FACT
 15      CONTINUE
         FX=1.0-((LXN+1.)*EXP(-LX))
         DIF=F-FX
         ADIF=ABS(DIF)
         SLOPE=(L*(LX**NN)*EXP(-LX))/FN
         IF(DIF.GT.0) XLO=X
         IF(DIF.LE.0) XHI=X
         X=X+DIF/SLOPE
         IF((X.LT.XLO).OR.(X.GT.XHI)) X=0.5*(XLO+XHI)
         WRITE(9,101) F,FX,DIF,SLOPE,XLO,X,XHI
 101  FORMAT(' F,FX,DIF,SLOPE,XLO,X,XHI:',3F11.7/20X,4F11.7)
      IF(ADIF.GT.EPSILN) GO TO 10
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE WEID(CUMP,IP,RP,PARTV,LIST,N,NCELL,NOF)
C  THIS SUBROUTINE FINDS THE PARTITIONS VALUES FOR DIVIDING
C THE SAMPLE INTO CELLS, BASED ON THE CUMULATIVE PROB
C VECTOR CUMP THE PARTITION VECTOR PARTV IS NCELL-1 LONG
      REAL LIST(400),CUMP(80),PARTV(80),RP(50),LAMBDA,
     * APOW,ALPH,F,X
      INTEGER N,NCELL,IP(50)
      WRITE(9,524) IP(2)
 524  FORMAT(1X,'WEIBULL DISTRIBUTION,',I3,
     *' PARAMETERS ESTIMATED')
      LAMBDA=RP(2)
      APOW=1.0/RP(1)
      NCM1=NCELL-1
      DO 20 I=1,NCM1
        F=CUMP(I)
        X=((ALOG(1.0-F))**APOW)/(-LAMBDA)
        PARTV(I)=X
 20   CONTINUE
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE WILCOX(LIST,LPTR,IP,RP,NOF)
C   THIS USES RANQUE TO RANK THE ABSOLUTE VALUES OF
C THE 1ST TREATMENT MINUS THE 2D TREATMENT, THEN SUMS THE
C RANKS OF THE POS DIFFERENCES AND THE NEG DIFFERENCES
C TO GET A T VALUE.  THE VAR SHIFT ALLOWS A CONSTANT TO BE
C ADDED TO THE 1ST TREATMENT, E.G., TO SEE IF THE 1ST TREAT
C PLUS A CONSTANT IS SIGNIFICANTLY LARGER THAN THE 2D TREAT
      REAL LIST(400),RP(50),ADIF(200),SVALS(400),RANK(400),
     * SRPOS,SRNEG,SRANK,SRSQ,P,PR,Z,ALPH1,ALPH2,ALFMIN,
```

```
     *CIF,ALPHAT,SHIFT
      INTEGER LPTR(17),KEY(400),IP(50),NOF,Q(200),
     * NNEG,NPOS,NODIF,POSDIF(200),IR
      SRANK=0.0
      SRSQ=0.0
      EPSILN=0.0001
      IR=0
      NNEG=0
      SRPOS=0.0
      SRNEG=0.0
      NPOS=0
      NVALS=LPTR(3)-1
      NBLOCK=LPTR(2)-1
      SHIFT=RP(1)
      DO 10 I=1,NBLOCK
         K=I+NBLOCK
         DIF=LIST(K)+SHIFT-LIST(I)
         ABDIF=ABS(DIF)
         IF(ABDIF.LT.EPSILN) GO TO 10
         IR=IR+1
         IF(DIF.LE.0) GO TO 13
             NPOS=NPOS+1
             POSDIF(IR)=1
13       IF(DIF.GE.0) GO TO 15
             NNEG=NNEG+1
             POSDIF(IR)=-1
15       ADIF(IR)=ABS(DIF)
10    CONTINUE
      NODIF=NBLOCK-NPOS-NNEG
      CALL RANQUE(IR,ADIF,SVALS,RANK,Q,KEY,1,NBLOCK,1)
      WRITE(9,211) (ADIF(I),I=1,IR)
      WRITE(9,211) (RANK(I),I=1,IR)
211   FORMAT(1X,'DIF/RNK:',7F8.2)
      DO 20 I=1,NBLOCK
         P=FLOAT(POSDIF(I))
         IF(P.GT.0.1) SRPOS=SRPOS+RANK(I)
         IF(P.LT.-0.1) SRNEG=SRNEG+RANK(I)
         PR=P*RANK(I)
         SRANK=SRANK+PR
         SRSQ=SRSQ+PR*PR
         ADIF(I)=ADIF(I)*P
20    CONTINUE
      WRITE(NOF,101) NBLOCK,SHIFT,NPOS,NNEG,NODIF,SRPOS,
     *    SRNEG
101   FORMAT(10X,'**WILCOXON SIGNED RANK TEST**'/1X,
     * 'IN',I4,' PAIRS,'/1X,'WITH 1ST VARIABLE SUBTRACTED',
     *' FROM THE 2D PLUS A SHIFT OF',F9.3/1X,' THERE ARE',
     *  I4,' POSITIVE DIFFERENCES,',I4,' NEGATIVES, AND'/
     *  5X,I4,' TIE(S)'/1X,'SUM OF POSITIVE RANKS:',F10.2/
     *  1X,'SUM OF NEGATIVE RANKS:',F10.2)
      IF (NBLOCK.LT.20) GO TO 50
      Z=SRANK/(SRSQ**0.5)
      CALL NORMV(1,Z,ALPH1)
      ALPH2=1.0-ALPH1
      ALFMIN=ALPH1
      IF(ALPH1.GT.ALPH2) ALFMIN=ALPH2
      IF(HYP.EQ.1) ALPHAT=2.0*ALFMIN
      IF(HYP.EQ.2) ALPHAT=ALPH1
      IF(HYP.EQ.3) ALPHAT=ALPH2
      ALPHYP=1.0-RP(2)
      WRITE (NOF,102) SRANK,SRSQ,Z,ALPHAT,ALPHYP
102   FORMAT(1X,'SUM OF SIGNED RANKS:',F10.2/
     *' SUM OF SQUARED SIGNED RANKS:',F10.2/'   T:',F10.3/
     *' ALPHA HAT:',F10.3,3X,'HYPOTHESISED ALPHA:',F10.3)
      IF(HYP.EQ.1) WRITE (NOF,131)
      IF(HYP.EQ.2) WRITE (NOF,132)
      IF(HYP.EQ.3) WRITE (NOF,133)
```

```
131   FORMAT(1X,'HO: E(X)=E(Y), H1: E(X)<>E(Y)')
132   FORMAT(1X,'HO: E(X)>=E(Y), H1: E(X)<E(Y)')
133   FORMAT(1X,'HO: E(X)<=E(Y), H1: E(X)>E(Y)')
      IF(ALPHYP.GT.ALPHAT) WRITE (NOF,121)
      IF(ALPHYP.LE.ALPHAT) WRITE (NOF,122)
121   FORMAT(1X,' REJECT THE NULL HYPOTHESIS')
122   FORMAT(1X,' DO NOT REJECT THE NULL HYPOTHESIS')
      GO TO 95
50    WRITE(NOF,103)
      CALL RANQUE(NBLOCK,ADIF,SVALS,RANK,Q,KEY,1,NBLOCK,1)
      WRITE(NOF,104) (SVALS(I),I=1,NBLOCK)
103   FORMAT(1X,'BECAUSE OF SMALL SAMPLE SIZE, TABLES '/
     * 1X,' SHOULD BE USED.  THE DIFFERENCES (NOT THE'/
     * 1X,'ABSOLUTE DIFFERENCES) FOLLOW IN RANK ORDER:')
104   FORMAT(1X,5F10.3)
 95   RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE MEDIA (NTREAT,LIST,LPTR,NIP,IP,NOF)
C  THIS SUBROUTINE FINDS THE MEDIAN VALUE OF A NUMBER OF
C VARIABLES/TREATMENTS, FINDS CELL COUNT OF #ABOVE/BELOW
C GRAND MEDIAN IN EACH TREAT, AND PERFORMS A R*C CONTINGENCY
C TEST ON THAT CELL COUNT MATRIX, USING FORM P.172 CONNOVER
C IN ORDER TO ALLOW GROUPING OF VARIABLES INTO AGGREGATE
C VARIABLES; THE AGVARI VECTOR CONTAINS THE INDEX OF AGVAR
C FOR EACH VAR (E.G.,VAR 1&3 IN TO AGVAR 1, VAR 2,4,5 INTO
C AGVAR 2; THE AGVARI IS 1,2,1,2,2). THIS INFORMATION IS IN
C CONTAINED IN THE IP VECTOR. NAGVAR IS THE NUMBER OF
C THE NUMBER OF AGGREGATE VARIABLES(E.G., HERE, 2)
C IF AGGREGATE VARIABLES ARE NOT USED,NIP=NUMBER OF INTEGER
C VARIABLES=0, NAGVAR=NTREAT= NUMBER
C OF VARIABLES IN DATA SET; AGVARI IS 1,2,...,NTREAT
C NAV(I) IS THE NUMBER OF ELEMENTS IN AGVAR(I)
C NA(I) IS # ELMS ABOVE MEDIAN IN AGVAR(I);NB(I)=#BELOW
C Q50 IS GRAND MED;
      INTEGER MED1,MED2,NB(15),NA(15),LL,UL,DF,NAV(17),
     *LPTR(17),KEY(400),AGVARI(17),NAGVAR,IP(17),NIP
     *A,B,N,NTREAT,NN,AN,AB
      REAL LIST(400),V,NUM,T,CONST,SVAL(400),Q50
      A=0
      B=0
      T=0
      IF (NIP.GT.0) GO TO 3
          NAGVAR=NTREAT
          DO 2  I=1,NTREAT
                AGVARI(I)=I
                NAV(I)=0
2         CONTINUE
          GO TO 10
3     CONTINUE
          NAGVAR=0
          DO 4  I=1,NTREAT
                AGVARI(I)=IP(I)
                IF (IP(1).GT.NAGVAR) NAGVAR=IP(1)
                NAV(I)=0
4         CONTINUE
10    DO 15 I=1,NAGVAR
          NB(I)=0
          NA(I)=0
15     CONTINUE
      N=LPTR((NTREAT+1))-1
      MED1=IFIX(0.5*FLOAT(N+1))
      MED2=IFIX(0.5*FLOAT(N+2))
      DO 18 I=1,N
          KEY(I)=I
          SVAL(I)=LIST(I)
18    CONTINUE
```

```
      CALL SHSCRT(SVAL,KEY,N)
      Q50=0.5*(SVAL(MED1)+SVAL(MED2))
C     WRITE(NOF,111) MED1,MED2,SVAL(MED1),SVAL(MED2),Q50
C111  FORMAT(1X,'MEC1,MED2',2I5,'SVAL(1,2),MEDIAN',3F10.3)
      DO 30 I=1,NTREAT
         LL=LPTR(I)
         UL=LPTR((I+1))-1
         K=AGVARI(I)
         NAV(K)=NAV(K)+UL-LL+1
         DO 20 J=LL,UL
               V=LIST(J)
               IF(V.GT.Q50) NA(K)=NA(K)+1
               IF(V.LE.Q50) NB(K)=NB(K)+1
 20      CONTINUE
 30   CONTINUE
      CO 35 I=1,NAGVAR
         A=A+NA(I)
         B=B+NB(I)
 35   CONTINUE
      NN=N*N
      AB=A*B
      CONST=FLCAT(NN)/FLOAT(AB)
      CO 50 I=1,NAGVAR
        IF (NAV(I).EQ.0) GC TO 50
         AN=A*NAV(I)
         NUM=(FLCAT((NA(I)))-FLOAT(AN)/FLOAT(N))**2
         T=T+CONST*NUM/FLOAT(NAV(I))
 50   CONTINUE
      DF=NAGVAR-1
      WRITE(NOF,101) Q50,NAGVAR
 101  FORMAT(1X,'THE MECIAN VALUE CF THE TEST IS',F10.3/1X,
     *'THE CELL COUNTS OF THE',I3,
     *' TREATMENTS CR AGGREGATES ARE:')
      WRITE (NCF,102) (I,I=1,NAGVAR)
 102  FORMAT(13X,16I4)
      WRITE(NOF,104)(NA(I),I=1,NAGVAR),A
 104  FORMAT(1X,' > MEDIAN    ',16I4)
      WRITE(NOF,103)(NB(I),I=1,NAGVAR),B
      WRITE(NOF,105) (NAV(I),I=1,NAGVAR),N
 103  FORMAT(1X,' =<    MEDIAN',16I4)
 105  FORMAT(1X,' TCTALS      ',16I4)
      WRITE (NCF,106) T,DF
 106  FORMAT(1X/1X,'CHI-SQUARED VALUE',F10.3/
     *1X,'CEGREES OF FREEDCM',I5)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE KRLWAL(LIST,LPTR,IP,RP,NOF,NTREAT,TREAT)
C          THIS SUMS THE RANKS OF EACH TREATMENT AND
C THE SQUARE OF ALL RANKS TC FIND THE OVERALL CHI SQUARE
C STATISTIC FOR THE NULL HYPOTHESIS THAT ALL TREATMENTS
C HAVE AN EQUAL MEAN.  IF REJECT THAT NULL, THEN THE
C ADDITIONAL NUMBERS COMPUTED HAVE SOME MEANING
      REAL TERM2,R,SSR,SSQ,T,LIST(400),RP(50),RANK(400),
     *   SRIDNI,C(20C),FN(17),SVALS(400),RNI(15)
      INTEGER LPTR(17),IP(50),NCF,TREAT(15),NTREAT,NJ(17),
     * N,UL,LL,KEY(400),DF1,DF2
      N=LPTR((NTREAT+1))-1
      DF1=N-1
      CF2=N-NTREAT
      CALL RANCUE(N,LIST,SVALS,RANK,Q,KEY,NTREAT,0,1)
      CO 10 I=1,NTREAT
         RNI(I)=0.0
 10   CONTINUE
      SSR=0.0
      SRIDNI=0.0
      DO 20 I=1,NTREAT
```

```
            LL=LPTR(I)
            UL=LPTR((I+1))-1
            NJ(I)=UL+1-LL
            FN(I)=FLOAT(NJ(I))
            DO 30  J=LL,UL
                 R=RANK(J)
                 RNI(I)=RNI(I)+R
                 SSR=SSR+R*R
 30         CONTINUE
            SRIDNI=SRIDNI+(RNI(I)*RNI(I))/FLOAT(NJ(I))
 20      CONTINUE
         TERM2=FLOAT(N*(N+1)*(N+1))/4.0
         SSQ=(SSR-TERM2)/FLOAT(N-1)
         T=(SRIDNI-TERM2)/SSQ
         WRITE (NOF,101) N,NTREAT, SSR
 101     FORMAT(10X,'**KRUSKAL-WALLIS TEST**'/1X,'IN SET OF',
        *I5,' DATA ELEMENTS, DIVIDED INTO',I5,' SAMPLES'/1X,
        *'THE SUM OF SQUARED RANKS WAS',F10.3,
        *' THE SUMS OF RANKS/ SAMPLE:'/
        */1X'  VAR NR',5X,'NR DATA ELEMS',5X'SUM OF RANKS')
         DO 40 I=1,NTREAT
         WRITE (NOF,102) TREAT(I),NJ(I),RNI(I)
 102     FORMAT(1X,I5,8X,I5,12X,F8.2)
 40      CONTINUE
         WRITE(NOF,103) T,DF1,DF2
 103     FORMAT(1X/1X,'YIELD A CHI SQUARE VALUE OF:',F10.3/1X,
        * 'WITH',I5,' DEGREES OF FREEDOM.  IF AND ONLY IF'/1X,
        * 'NULL HYPOTHESIS REJECTED, FOL NUMBERS CAN'/1X,
        * 'TEST WHETHER SPECIFIC PAIRS OF VARIABLES HAVE'/1X,
        * 'THE SAME MEAN. IF ABS DIFFERENCE IN MEAN RANKS'/1X,
        * '|R(I)/N(I)-R(J)/N(J)| > POOLED STANDARD DEVIATION'/
        *1X,'TIMES THE T QUANTILE 1-ALPHA/2 THEN REJECT THE'/
        *1X,'HYPOTHESIS THAT THE PAIR OF MEANS ARE EQUAL'/1X,
        * '  (DEGREES OF FREEDOM HERE:',I4,')'//3X,'I  ',
        *' J    ABS DIF MEAN RANK    POOLED STANDARD DEVIATION')
         CONST=(SSQ*(FLOAT(N)-1.0-T)/FLOAT(DF2))**0.5
         NTM1=NTREAT-1
         DO 60 I=1,NTM1
            IP1=I+1
            DO 70  J=IP1,NTREAT
                 ADMR=ABS((RNI(I)/FN(I))-(RNI(J)/FN(J)))
                 SDPOOL=CONST*((1.0/FN(I))+(1.0/FN(J)))
                 WRITE(NOF,104) I,J,ADMR,SDPOOL
 104             FORMAT(2X,2I3,6X,F8.3,14X,F10.3)
 70         CONTINUE
 60      CONTINUE
         RETURN
         END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
         SUBROUTINE SQLARK (LIST,LPTR,NTREAT,TREAT,IP,RP,NOF)
C  THIS TESTS WHETHER A DATA SET HAS EQUAL VARIANCE AMONG
C ALL VARIABLES BY (1) FINDING THE ABSOLUTE DEVIATIONS OF
C EACH VARIABLE'S ELEMENTS, IN SUBROUTINE ADEV, (2) RANKING
C THOSE DEVIATIONS (RANQUE), AND (3) SUMMING,SQUARING,ETC.,
C THOSE RANKS AS REQUIRED TO FIND THE TEST STATISTIC
         REAL LIST(400),DEV(400),RP(50),FN,FM,FNT,R(400),
        *SUMRJ(17),FNJ(17),R2MU,RMU,T1,T2,NUM,RSQ,SSSR2J,
        *SUMR2J(17)
         INTEGER LPTR(17),NTREAT,TREAT(17),IP(50),NOF,KEY(400)
         NT=LPTR((NTREAT+1))-1
         FNT=FLOAT(NT)
         CALL ADEV(LIST,LPTR,NTREAT,DEV)
         DO 3 I=1,NT
            KEY(I)=I
 3       CONTINUE
         CALL RNQ(DEV,KEY,NT,R)
         DO 5 J=1,NTREAT
```

```
         SUMR2J(J)=C
         SUMRJ(J)=0
5     CONTINUE
      SUMR=0.0
      SUMR2=0.0
      SUMR4=0.C
      J=1
      CO 10 I=1,NT
         RSQ=R(I)*R(I)
         SUMR=SUMR+F(I)
         SUMR2=SLMR2+RSQ
         SUMR4=SLMR4+RSC*RSQ
         INCRJ=LPTR((J+1))
         IF(I.GE.INCRJ) J=J+1
               SUMRJ(J)=SUMRJ(J)+R(I)
               SLMR2J(J)=SUMR2J(J)+RSQ
10    CONTINUE
      R2MU=SUMR2/FNT
      RMU=SUMR/FNT
      IF (NTREAT.GT.2) GO TC 40
         FN=FLCAT(LFTR(2)-1)
         FM=FLCAT(LFTR(3)-LPTR(2))
         DTERM2=FN*FM/(FNT-1.0)
         DTERM1=CTEFM2/FNT
         DENOM=(CTEFM1*SUMR4-DTERM2*R2MU*R2MU)**0.5
         NUM=SUMR2J(1)-FN*R2MU
         T1=NUM/CENCM
         WRITE(9,101) DTERM1,DTERM2,SUMR4,R2MU,SUMR2J(1)
101      FORMAT(1X,'DTERM1-2,SUMR4,R2MU,SUMJ2.1',5F9.4)
         WRITE(NCF,102) T1
102      FCRMAT(10X,'**** SQUARED RANK TEST FOR ',
    * 'EQUAL VARIANCE ****'/1X,'TEST STATISTIC ',
    * 'FOR THIS TWC SAMPLE DATA SET IS',F10.4)
      GO TO 95
40    CONTINUE
         DSQ=(SLMR4-FNT*R2MU*R2MU)/(FNT-1.0)
         SSJMU=C.0
         DC 50 J=1,NTREAT
               SSSR2J=SUMR2J(J)*SUMR2J(J)
               SSJMU=SSJMU+SSSR2J/FLOAT(LPTR((J+1))-LPTR(J))
50       CONTINLE
         T2=(SSJMU-FNT*R2MU*R2MU)/DSQ
      WRITE(9,103) SSJMU,R2MU,SUMR4,DSQ
103   FORMAT(1X,' SSJUM,R2MU,SUMR4,DSQ',2F10.4,F20.4,F10.4)
      WRITE(NOF,104) NTREAT, T2
104   FORMAT(1CX,
    * '**** SQUAREC RANK TEST FOR EQUAL VARIANCE ****'/
    *    ' IN A CATA SET OF ',I3,' VARIABLES'/1X,
    *    ' THE 1EST STATISTIC:',F10.4)
95    RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE ADEV(LIST,LPTR,NTREAT,DEV)
      REAL LIST(400),RP(50),DEV(400),VARX,MU,X(200)
      INTEGER LPTR(17),K,NTREAT,IP(50),NOF,LL,IP1,NI
      CO 10 I=1,NTREAT
         IP1=I+1
         LL=LPTR(I)
         NI=LPTF(IP1)-LL
         DO 12 K=1,NI
               X(K)=LIST((LL+K-1))
12       CONTINLE
         CALL MLVAR(X,NI,MU,VARX)
         DO 30 K=1,NI
               J=LL+K-1
               DEV(J)=ABS(LIST(J)-MU)
30       CCNTINUE
```

```
         LI=LPTR(I)
         WRITE(9,105) I,LI,LIST(LI),MU,DEV(LI)
 105  FORMAT(1X,'VAR#,LPTR;LIST,MU,DEV OF LPTR:',2I3,3F10.4)
 10   CONTINUE
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-//--/-/-/-/-
      SUBROUTINE HARTLE(LIST,LPTR,NTREAT,TREAT,IP,RP,NOF)
C  THIS SUBROUTINE FINDS THE STATISTIC FOR THE HARTLEY'S
C TEST FOR EQUAL VARIANCE, AN F RATIO BETWEEN THE SMALLEST
C AND LARGEST VARIANCE IN A SET OF DATA.
      REAL LIST(400),RP(50),X(200),VMIN,VMAX,VARX,FRATIO,MU
      INTEGER LPTR(17),NTREAT,TREAT(16),IP(50),NOF,LL,UL,
     *IP1,NI
      VMIN=100000.0
      VMAX=-10.0
      DO 10 I=1,NTREAT
         IP1=I+1
         LL=LPTR(I)
         NI=LPTR(IP1)-LL
         DO 12 K=1,NI
            X(K)=LIST((LL+K-1))
 12      CONTINUE
         CALL MUVAR(X,NI,MU,VARX)
         IF (VARX.GE.VMIN) GO TO 20
            VMIN=VARX
            IVMIN=TREAT(I)
 20      IF(VARX.LE.VMAX) GO TO 10
            VMAX=VARX
            IVMAX=TREAT(I)
 10   CONTINUE
      FRATIO=VMAX/VMIN
      WRITE(NOF,101) NTREAT,IVMIN,VMIN,IVMAX,VMAX,FRATIO
 101  FORMAT(10X,
     * '***** HARTLEY TEST FOR EQUAL VARIANCE****'/2X,'IN',
     *' A DATA SET WHICH COMPARED ',I3,' VARIABLES, '/1X,
     *' VARIABLE',I3,' HAD THE SMALLEST VARIANCE:',F10.4,
     *','/1X,' WHILE VARIABLE',I3,' HAD THE LARGEST:',F10.4,
     */1X,' WHICH GIVES AN F RATIO OF:',F10.4)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE CORREL(LIST,LPTR,NTREAT,TREAT,NOF,IP,RP)
C  THIS SUBROUTINE TAKES LIST, COMPOSED OF TWO EQUAL LENGTH
C VARIABLES, PERFORMS PEARSON'S RHO ON THEM, THEN RANKS
C EACH VARIABLE, PERFORMS PEARSON'S RHO ON EACH TO YIELD
C SPEARMAN'S RHO, AND IF REQUESTED, FIGURES KENDALL'S TAU
C IP(1) IS OPTION:-1=SPEARMAN ONLY,0=PEARSON&SPEARMAN,
C             1=PEARSON,SPEARMAN AND KENDALL
      REAL LIST(400),RP(50),SVALS(200),RANK(400),B1,B0,TAU,
     *   TRANKS(200),TVALS(200),SRANQ(200),TRANQ(200)
      INTEGER LPTR(17),KEY(400),NTREAT,TREAT(17),NOF,
     *   TKEY(200),SKEY(200),OPT,IP(50)
      NPAIR=LPTR(2)-1
      IF (IP(1).LT.0) GO TO 5
      WRITE(NOF,201)
 201  FORMAT(10X,'**CORRELATION**')
      CALL RHOER(LIST,NPAIR,RHO,B1,B0,0)
      WRITE(NOF,101) RHO
 101  FORMAT('PEARSON"S CORRELATION ON THE',
     * ' ACTUAL VALUES OF TWO VARS:'/10X,F10.3)
 5    DO 10 I=1,NPAIR
         K=I+NPAIR
         SKEY(I)=I
         TKEY(I)=I
         SVALS(I)=LIST(I)
         TVALS(I)=LIST(K)
```

```
 10    CONTINUE
       CALL  RNQ  (SVALS,SKEY,NPAIR,SRANQ)
       CALL  RNQ  (TVALS,TKEY,NPAIR,TRANQ)
       DO 15 I=1,NPAIR
          K=I+NPAIR
          RANK(I)=SRANQ(I)
          RANK(K)=TRANQ(I)
 15    CONTINUE
       CALL  RHOER(RANK,NPAIR,RHO,B1,B0,0)
       WRITE(NOF,102) RHC
 102   FORMAT(10X,'**CORRELATION**'/1X,'PEARSON"S ',
     * 'CORRELATICN ON THE RANKS OF TWO VARS',
     */1X,', I.E.,SPEARMAN"S RHC:',F10.3)
       IF (IP(1).LT.1) GO TO 95
       NDIS=0
       NCON=0
       CO 20 I=1,NPAIR
          IS=SKEY(I)
          IP1=I+1
          ISP1=IS+1
          NCONIS=C
          NDISIS=0
          DC 25 L=IP1,NPAIR
                J=SKEY(L)
                SCIF=SRANC(J)-SRANQ(IS)
                IF(SDIF.LT.0.1) GO TC 25
                     TDIF=TRANQ(J)-TRANQ(IS)
                     IF(TCIF.LT.0.1) GO TO 28
                          NCON=NCCN+1
                          NCONIS=NCONIS+1
                          GO TO 25
 28                  IF(TCIF.GT.-0.1) GO TO 25
                          NDIS=NDIS+1
                          NDISIS=NDISIS+1
 25             CCNTINUE
          WRITE(9,141) IS,NCONIS,NDISIS
 141      FORMAT(1X,'INDEX OF X=',I5,
     *         ' #CONCORDANT,DISCCNCORDANT',2I5)
 20    CONTINUE
       TAU=2.0*FLCAT(NCON-NDIS)/FLOAT(NPAIR*(NPAIR-1))
       WRITE(NOF,142) NCCN,NDIS,TAU
 142   FORMAT(10X,'**** KENCALLS TAU ****'/1X,
     *' TOTAL NUMBER OF CONCORDANT PAIRS:',I5/1X,
     *' TOTAL NUMBEF OF DISCONCORDANT PAIRS:',I5/1X,
     *' TAU=',F10.4)
 95    RETURN
       END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
       SUBROUTINE REGRET(LIST,LPTR,IP,RP,TREAT,NCF)
C  THIS USES RHCER TO FIND THE LEAST SQUARES FIT UNLESS
C USER SPECIFIES THE HYPOTHESIZED LINE FOR THE REGRESSION
C BETWEEN AN INCEP (EXPLANATORY) VARIABLE AND A DEPENDENT
C VARIABLE.  IT TESTS FIT USING THE SPEARMANS RHO TC FIND
C RANK CORRELATICN BETWEEN INDEPENDENT VARIABLE AND THE
C RESIDUALS.  IF USER REQ'D IT, IT WILL FINC CONFIDENCE
C BOUNCS ON THE SLOPE (B1) OF THE FITTED LINE.
C IP(1):INDEX CF DEFENDENT VARIABLE IN LIST (1 CR 2)
C IP(2): 0=USE LEAST SQUARES EST B1,
C        1=HYP B1; SUPPLIED IN RP(1)
C IP(3): 0=NO EST'S OF Y BY X,ELSE = # OF Y'S EST,RP(3+)
C IP(4): 0=NO CCNFICENCE INTERVALS, 1=YES
C RP(2): VALUE CF W TO COMPUTE THE SLOPE INDICES
       INTEGER LPTR(17),IP(50),NOF,NPAIR,OPTION,TREAT(15),
     *   NPM1,KEY(20C),NR,NS,DIP(4),NESTY,IX,L,K,I17,I47
       REAL LIST(400),RP(50),B1,B0,RHO,S(20000),DENOM,ADEM,
     *X,Y,XSUM,YSUM,EY,UX(400),QALFD2,B1R
       WRITE(NOF,201)
```

```
201   FORMAT(10X,'**NON-PARAMETRIC REGRESSICN**')
      EPSILN=0.CC01
      CPTICN=IF(1)
      CIP(1)=-1
      XBAR=0
      YBAR=0
      IX=3-OPTICN
      IY=OPTION
      NPAIR=LPTR(2)-1
      CALL RHOER(LIST,NPAIR,RHO,B1,B0,CPTICN)
      WRITE (NCF,101) TREAT(IY),TREAT(IX),B1,B0
101   FORMAT(1X,
     *' WITH DEPENDENT VARIABLE AS VARIABLE #',I5/1X,
     *' AND EXPLANATORY/INDEPENDENT VARIABLE AS #',I5/1X,
     *' THE LEAST SCUARES FIT OF LINEAR REGRESSION IS:'/1X,
     *' DEP VAR = ',F9.3,' * INDEP VAR + ',F10.3)
      B1R=RP(1)
      IF ( IP(2).LT.1) B1R=B1
      IXL=LPTR(IX)-1
      IYL=LPTR(IY)-1
      WRITE(9,678) IXL,IYL
678   FORMAT(1X,' X & Y INDICES:',2I5)
      CO 20 I=1,NPAIR
         J=NPAIR+I
         X=LIST(IXL+I)
         UX(J)=X
         Y=LIST(IYL+I)-B1R*X
         UX(I)=Y
      WRITE(9,987) LX(I),UX(J)
987   FORMAT(1X,'X, RESID:',2F10.4)
20    CONTINUE
      WRITE(NOF,102) B1R
102   FORMAT(1X,'TO TEST CORRELATICN BETWEEN THE INCEPEN',
     *'DENT VARIABLE'/1X,'AND THE RESIDUALS OF THE FIT,'/
     *1X,' WHERE B1=',F1C.4)
      CALL CORREL(UX,LPTR,NTREAT,TREAT,NOF,DIP,RP)
      NESTY=IP(3)
      IF(NESTY.EC.0) GO TO 24
      CO 21 I=1,NPAIR
         XSUM=XSLM+LIST(IXL+I)
         YSUM=YSLM+LIST(IYL+I)
21    CONTINUE
      IF(IP(2).EQ.1) B0=(YSUM-B1*XSUM)/FLOAT(NPAIR)
      WRITE(NOF,121) NESTY,B1
121   FORMAT(1X,'ESTIMATE OF',I3,' DEPENDENT VARIABLES BY ',
     *'REGRESSICN,'/1X,'WITH B1=',F10.4)
      CO 22 I=1,NESTY
         EY=B1*RP((2+I))+B0
         WRITE(NCF,122) RP((2+I)),EY
122      FORMAT(1X,' INCEP VAR VALUE:',F10.4,
     *   ', DEP VAR ESTIMATE:',F10.4)
22    CONTINUE
24    IF(IP(4).EC.0) GO TO 95
         IS=0
         QALFD2=1.0-((1.0-RP(2))*0.5)
         N=NPAIR
         IF(N.GT.60)
     *      W=QALFD2*((FLOAT(N*(N-1)*(2*N+5))/18.0)**0.5)
         IF(N.LE.60) W=RP(2)
         NPM1=NPAIR-1
         DO 30 I=1,NPM1
            LL=I+1
            CO 35 J=LL,NPAIR
               CENOM=LIST(IXL+I)-LIST(IXL+J)
               ADEM=ABS(DENOM)
               IF(ADEM.LT.EPSILN) GO TO 35
                  IS=IS+1
```

```
                       S(IS)=(LIST(IYL+I)-LIST(IYL+J))/DENOM
                       KEY(IS)=IS
 35            CONTINUE
 30      CONTINUE
         CALL SHSORT(S,KEY,IS)
         I17=17
         I47=47
          WRITE(9,876) S(I17),S(I47),IS
 876      FORMAT(1X,'SLOPES, #SLOPES',2F9.4,I5)
         NR=IFIX(0.5*(IS-W))
         NS=IFIX(0.5*(IS+W)+1.96)
         WRITE(NOF,103) W,NR,S(NR),NS,S(NS)
 103     FORMAT(1X,' BASED ON A VALUE OF W=',F10.4/1X,
     *'INDICES OF CONFIDENCE INTERVAL, & VALUES OF SLOPE:'/
     *   1X,'LOWER BOUND INDEX:',I3,' WITH VALUE:',F10.4/1X,
     *   ' UPPER BOUND INDEX:',I3,' WITH VALUE:',F10.4)
 95      RETURN
         END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE MONORE(LIST,LPTR,IP,RP,TREAT,NOF)
C  THIS SUBROUTINE PERFORMS MONOTONE REGRESSION BASED ON THE
C LEAST SQUARED FIT OF THE RANKS OF 2 VARIABLES.  IF THE
C USER DESIRES, ESTIMATES OF SPECIFIC VALUES OF DEPENDENT
C VARIABLE Y BASED ON VALUES OF X CONTAINED IN RP VECTOR.
C ALSO, IF THE USER DESIRES, A TRACE OF THE REGRESSION WILL
C BE LISTED.
C IP(1): # OF DEPENDENT VARIABLE (1 OR 2)
C IP(2): # OF SPECIFIED X VALUES FOR EXTIMATION OF Y VALUES
C IP(3): DO YOU WANT A TRACE? 1=YES 0=NO
      REAL LIST(400),X(200),Y(200),RX(200),RY(200),TERM2,
     * B2,A2,FN,RP(50),SXX,SXY
      INTEGER KEYX(200),KEYY(200),NPAIR,IX,IY,IP(50),
     *LPTR(17),TREAT(17)
      NPAIR=LPTR(2)-1
      FN=FLOAT(NPAIR)
      IX=(2-IP(1))*NPAIR
      IY=(IP(1)-1)*NPAIR
      WRITE(9,210) IX,IY
 210  FORMAT(1X,' INTO MONORE;INDEX OF FIRST X,Y:',2I5)
      DO 10 I=1,NPAIR
         X(I)=LIST(I+IX)
         Y(I)=LIST(I+IY)
         KEYX(I)=I
         KEYY(I)=I
 10   CONTINUE
      CALL RNQ(X,KEYX,NPAIR,RX)
      CALL RNQ(Y,KEYY,NPAIR,RY)
      SXY=0.0
      SXX=0.0
      DO 15 I=1,NPAIR
         X(I)=LIST(I+IX)
         Y(I)=LIST(I+IY)
 15   CONTINUE
      DO 20 I=1,NPAIR
         SXY=SXY+RX(I)*RY(I)
         SXX=SXX+RX(I)*RX(I)
 20   CONTINUE
      TERM2=FN*((FN+1.0)**2)/4.0
      B2=(SXY-TERM2)/(SXX-TERM2)
      A2=(1.0-B2)*(FN+1.0)/2.0
      WRITE(NOF,101) TREAT(IP(1)),TREAT((3-IP(1))),B2,A2
 101  FORMAT(10X,'**MONOTONE REGRESSION**'/1X,' USING ',
     *'LEAST SQUARES REGRESSION ON THE RANKS OF VARIABLES'/
     *1X,'WITH VARIABLE #',I3,' AS DEPENDENT VARIABLE Y'/
     *1X,'AND VARIABLE #',I3,' AS INDEPENDENT VARIABLE X'/
     *1X,' R(Y) = ',F9.3,' * R(X) +',F10.3)
      NEST=IP(2)
```

```
      N=NPAIR
      IF (NEST.GT.0)
     *   CALL ESTY(A2,B2,X,Y,KEYX,KEYY,RX,RY,N,NEST,RP,NOF)
      IF (IP(3).EQ.1)
     *   CALL TRACE(A2,B2,X,Y,RX,RY,KEYX,KEYY,N,NOF)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE ESTY
     * (A2,B2,X,Y,KEYX,KEYY,RX,RY,N,NEST,RP,NOF)
C  THIS FINDS THE ESTIMATES OF THE DEPENDENT VAR Y
C FOR VALUES OF THE INDEPENDENT VAR X
      REAL A2,B2,X(200),Y(200),RX(200),RY(200),RP(50)
      INTEGER KEYX(200),KEYY(200),KEY(200),NEST,N,XSTART,
     * YSTART
      IF (NEST.EQ.0) GO TO 95
      XSTART=0
      DO 5 I=1,NEST
         KEY(I)=I
 5    CONTINUE
      CALL SHSORT(RP,KEY,NEST)
      WRITE (9,211) RP(1),KEY(1),NEST
 211  FORMAT(1X,'INTO ESTY: RP(1),KEY(1),NEST:',F10.3,2I5)
      DO 10 I=1,NEST
         VALX=RP(I)
        CALL VTOR(X,RX,KEYX,VALX,RNKX,N,XSTART)
        IF(RNKX.GE.1) GO TO 15
          WRITE(NOF,101) I,VALX,X(KEYX(1)),X(KEYX(N))
 101      FORMAT(1X,'EST #',I3,' NOT POSSIBLE; X=',F10.4/
     *      9X,' FALLS OUTSIDE RANGE OF',F10.4,' TO',F10.4)
             GO TO 10
 15      RNKY=A2+B2*RNKX
         YSTART=0
         CALL RTOV (Y,RY,KEYY,VALY,RNKY,N,YSTART)
         WRITE(NOF,102) I,VALX,VALY
 102     FORMAT(1X,'EST #',I3,
     *              ': X=',F10.4,' => E(Y|X) OF',F10.4)
 10   CONTINUE
 95   RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE VTOR(Z,RZ,KEYZ,VALZ,RNKZ,N,ZSTART)
C        HAS A VALUE, FINDS ITS RANK
      REAL Z(200),RZ(200),RNKZ,VALZ
      INTEGER KEYZ(200),N,ZSTART
      RNKZ=-1.0
      IF((VALZ.LT.Z(KEYZ(1))).OR.(VALZ.GT.Z(KEYZ(N))))
     *        GO TO 95
      CALL INTERP(Z,RZ,KEYZ,VALZ,RNKZ,N,ZSTART)
      GO TO 97
 95   WRITE(9,101) VALZ,Z(KEYZ(1)),Z(KEYZ(N))
 101  FORMAT(1X,'OUT OF RANGE:VAL,LL,UL:',3F10.4)
 97   RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE RTOV(Z,RZ,KEYZ,VALZ,RNKZ,N,ZSTART)
C        HAS A RANK, FINDS ITS VALUE
      REAL Z(200),RZ(200),RNKZ,VALZ
      INTEGER KEYZ(200),N,ZSTART
      IF((RNKZ.GT.RZ(KEYZ(1))).AND.(RNKZ.LT.RZ(KEYZ(N))))
     *      GO TO 50
         IF(RNKZ.LE.RZ(KEYZ(1))) VALZ=Z(KEYZ(1))
         IF(RNKZ.GE.RZ(KEYZ(N))) VALZ=Z(KEYZ(N))
        GO TO 95
 50   CALL INTERP(RZ,Z,KEYZ,RNKZ,VALZ,N,ZSTART)
 95   RETURN
      END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE INTERP(A,B,KEY,AVAL,BVAL,N,ISTART)
C  THIS SUBROUTINE DOES LINEAR INTERPOLATION: KEY IS THE
C ORDER OF VALS A AND B.  AVAL IS KNOWN, FIND BVAL. ISTART
C IS THE INDEX OF KEY AT WHICH TO START LOOKING: WITH PRIOR
C KNOWLEGED, SUCH AS IF AVAL'S WERE SORTED, ISTART MAY BE
C GREATER THAN 0
      REAL A(200),B(200),AVAL,BVAL,PROP,BPROP
      INTEGER I,ISTART,N,KL,KU,KEY(200)
      I=ISTART
 10   CONTINUE
         I=I+1
         KU=KEY(I)
         IF((AVAL.GT.A(KU)).AND.(I.LT.N)) GO TO 10
      KL=KEY((I-1))
      PROP=(AVAL-A(KL))/(A(KU)-A(KL))
      BPROP=PROP*(B(KU)-B(KL))
      BVAL=B(KL)+BPROP
      ISTART=I-1
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE TRACE(A2,B2,X,Y,RX,RY,KEYX,KEYY,N,NOF)
C     THIS SUBROUTINE FINDS THE TRACE OF THE REGRESION
C IN GENERAL BY TAKING THE ACTUAL RANK OF Y, FINDING
C THE EXPECTED RANK OF X AND THE CORRESPONDING VALUE OF X,
C AND THEN PLOTTING THAT X WITH THE ACTUAL VALUE OF Y.
      REAL EX(200),EY(200),X(200),Y(200),RX(200),RY(200),
     *RYB,A2,B2,RYS
      INTEGER KEYX(200),KEYY(200),N,NOF
      K=0
      WRITE(NOF,101)
 101  FORMAT(10X,' TRACE OF THE MONOTONE REGRESSION '/1X,
     * '    X    ',6X,'   Y   ')
      RYS=A2+B2*(RX(KEYX(1)))
      IF(RYX.LT.RY(KEYY(1))) GO TO 5
         K=K+1
         EX(K)=X(KEYX(1))
         YSTART=0
         CALL RTCV (Y,RY,KEYY,YY,RYS,N,YSTART)
         EY(K)=YY
         WRITE(NOF,102) EX(K),EY(K)
 5    DO 10 I=1,N
         J=KEYY(I)
         ERX=(RY(J)-A2)/B2
         XSTART=0
         CALL RTCV (X,RX,KEYX,XX,ERX,N,XSTART)
         K=K+1
         EX(K)=XX
         EY(K)=Y(J)
         WRITE(NOF,102) EX(K),EY(K)
 10   CONTINUE
      RYB=A2+B2*(RX(KEYX(N)))
      IF(RYB.GT.RY(KEYY(N))) GO TO 20
         K=K+1
         EX(K)=X(KEYX(N))
         YSTART=0
         CALL RTCV (Y,RY,KEYY,YY,RYB,N,YSTART)
         EY(K)=YY
         WRITE(NOF,102) EX(K),EY(K)
 102  FORMAT(1X,2F10.4)
 20   CALL PLOTT(EX,EY,N,1)
      RETURN
      END
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
      SUBROUTINE KOLMOG (LIST,LPTR,TREAT,IP,RP,NOF)
C     THIS SUBROUTINE FINDS THE TEST STATISTIC FOR THE
```

```
C KOLMOGOROV STATISTIC FOR SAMPLES OF UNDER 31 ELEMENTS.
C DEPENDING ON NULL HYP, THE TEST STATISTIC IS THE LARGEST,
C SMALLEST OR GREATEST ABSOLUTE DIFFERENCE F(X)-S(X),
C WHERE F IS THE HYPOTHESISED THEORETICAL DIST AND
C S IS THE EMPIRICAL CDF.  IP(1) =  CHOICE OF DISTR;
C IP(2) IS THE CHOICE OF HYPOTHESIS
C  SEE THE DISTRIBUTION SUBROUTINES FOR MEANINGS OF THE
C  OTHER PARAMETERS RP AND IP.
      REAL LIST(400),S(400),F(400),RP(50),FN,TSTAT
      INTEGER LPTR(17),IP(50),NOF,IDIST,HYP,KEY(400)
      WRITE(NOF,101)
 101  FORMAT(10X,'**** KOLMOGOROV TEST ****')
      HYP= IP(2)
      N=LPTR(2)-1
      FN=FLOAT(N)
      IDIST=IP(1)
      DO 10 I=1,N
         S(I)=FLOAT(I)/FN
         KEY(I)=I
 10   CONTINUE
      CALL SHSORT(LIST,KEY,N)
      IF(IDIST.EQ.1) CALL NORMF(LIST,N,RP,F,NOF)
      IF(IDIST.EQ.2) CALL EXPF(LIST,N,RP,F,NOF)
      IF(IDIST.EQ.3) CALL UNIF(LIST,N,RP,F,NOF)
      IF(IDIST.EQ.4) CALL ERLF(LIST,N,RP,IP,F,NOF)
      IF(IDIST.EQ.5) CALL WEIBF(LIST,N,RP,F,NOF)
      CALL FINDT(S,F,N,TSUP,TMIN,TMAX,NOF)
      IF(HYP.EQ.1) CALL THYPA(TSUP,NOF)
      IF(HYP.EQ.2) CALL THYPB(TMAX,NOF)
      IF(HYP.EQ.3) CALL THYPC(TMIN,NOF)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE ERLF(LIST,N,RP,IP,F,NOF)
C THIS FINDS THE THEORETICAL ERALNG DISTRIBUTION F, WHERE
C NN=IP(1)=SHAPE PARAMETER, AND L=RP(1)=SCALE PARAMETER
      REAL LX,LXN,L,X,LIST(400),RP(50),F(400),DN
      INTEGER IP(50),I,N,FACT,NNM1
      NN=IP(1)
      NNM1=NN-1
      DO 30 I=1,N
        FACT=1
        X=LIST(I)
        LX=L*X
        LXN=0
        DO 23  J=1,NNM1
            FACT=FACT*J
            LXN=LXN+(LX**J)/FACT
 23     CONTINUE
        IF  (X.LE.0) F(I)=0
        IF  (X.GT.0) F(I)=1.0-(1.0+LXN)*EXP(-LX)
 30   CONTINUE
      WRITE(NOF,101) NN,L
 101  FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTUION ERLANG'/1X,
     *'WITH PARAMETERS N=',I5,' AND LAMBDA=',F10.4)
      RETURN
      END
C--------------------------------------------------------------
      SUBROUTINE WEIBF(LIST,N,RP,F,NOF)
C   THIS FINDS THE THEORETICAL WEIBULL DISTIBUTION F, WHERE
C  A=RP(1)=SHAPE PARAMETER, AND L=RP(2)=SCALE PARAMETER
C  LAMBDA
      REAL A,L,X,LIST(400),RP(50),F(400)
      INTEGER I
      A=RP(1)
      L=RP(2)
      DO 10 I=1,N
```

```
          X=LIST(I)
          IF(X.LE.0) F(I)=0
          IF(X.GT.0) F(I)=1.0-EXP(-((L*X)**A))
 10    CONTINUE
       WRITE(NOF,101) A,L
 101   FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTION WEIBULL'/1X,
      *'WITH PARAMETERS ALPHA=',F10.4,' AND LAMBDA=',F10.4)
       RETURN
       END
C-----------------------------------------------------------------
       SUBROUTINE UNIF(LIST,N,RP,F,NOF)
C THIS SUBROUTINE FINDS THE THEORETICAL UNIFORM DISTRIBUTION
C F, WHERE A=RP(1)=LOWER BOUND, B=RP(2)=UPPER BOUND
       REAL C,A,B,X,LIST(400),RP(50),F(400)
       INTEGER I
       A=RP(1)
       B=RP(2)
       C=B-A
       DO 10 I=1,N
          X=LIST(I)
          IF (X.LE.A) F(I)=0
          IF (X.GE.B) F(I)=1
          IF((X.GT.A).AND.(X.LT.B)) F(I)=(X-A)/C
 10    CONTINUE
       WRITE(NOF,101) A,B
 101   FORMAT(1X,'WITH HYPOTHESISED DISTRIBUTUION UNIFORM',
      */1X,'WITH PARAMETERS LOWER LIMIT=',F10.4,
      * ' AND UPPER LIMIT=',F10.4)
       RETURN
       END
C-/-/--/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--
       SUBROUTINE LILIEF (LIST,LPTR,TREAT,IP,RP,NOF)
C      THIS SUBROUTINE FINDS THE TEST STATISTIC FOR THE
C LILLIEFORS TEST FOR SAMPLES OF UNDER 31 ELEMENTS.
C DEPENDING ON HYPOTHESIS; TEST STATISTIC IS THE LARGEST
C SMALLEST OR GREATEST ABSOLUTE DIFFERENCE F(X)-S(X),
C WHERE F IS THE HYPOTHESISED THEORETICAL DISTRIBUTION AND
C S IS THE EMPIRICAL CDF.  IP(1) IS THE CHOICE OF
C DISTRIBUTIONS; IP(2) IS CHOICE OF HYPOTHESIS. PARAMETERS
C ARE ESTIMATED: RP(1)=SAMPLE MEAN,RP(2)=VARIANCE
C SEE THE DISTRIBUTION SUBROUTINES FOR THE MEANINGS OF THE
C OTHER PARAMETERS RP AND IP.
       REAL LIST(400),S(400),F(400),RP(50),FN,TSTAT,MU,
      *TMIN,TMAX,VAR,TSUP
       INTEGER LPTR(17),IP(50),NOF,IDIST,HYP,KEY(400)
       HYP=IP(2)
       N=LPTR(2)-1
       FN=FLOAT(N)
       CALL MUVAF(LIST,N,MU,VAR)
       WRITE(NOF,101) MU,VAR
 101   FORMAT(10X,'**** LILLIEFORS TEST ****'/1X,
      *'WITH COMPUTED SAMPLE MEAN=',F10.4,
      * ', AND VAR=',F10.4)
       RP(1)=MU
       RP(2)=VAR
       IDIST=IP(1)
       IF(IDIST.EQ.2) RP(1)=1.0/MU
       DO 10 I=1,N
          S(I)=FLOAT(I)/FN
          KEY(I)=I
 10    CONTINUE
       CALL SHSORT(LIST,KEY,N)
       IF(IDIST.EQ.1) CALL NORMF(LIST,N,RP,F,NOF)
       IF(IDIST.EQ.2) CALL EXPF(LIST,N,RP,F,NOF)
       CALL FINDT(S,F,N,TSUP,TMIN,TMAX,NOF)
       IF(HYP.EQ.1) CALL THYPA(TSUP,NOF)
       IF(HYP.EQ.2) CALL THYPB(TMAX,NOF)
```

```
      IF(HYP.EQ.3) CALL THYPC(TMIN,NOF)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE WILKSH (LIST,LPTR,TREAT,IP,RP,NOF)
C    THIS SUBROUTINE FINDS THE TEST STATISTIC FOR THE
C SHAPIRO-WILK TEST FOR NORMALITY, N < 31 ELEMENTS.
C IP(1)= NUMBER OF COMPARISONS= TRUNC(# DATA ELEMS/2)
C RP(1),....,RP(K)= WEIGHTS TO SCALE COMPARISONS
C THE DENOMINATOR IS THE SUM OF SQUARED DEVATIONS OF X;AND
C HERE IT IS FOUND BY SXX=(N-1)*VAR(X)
C THE NUMERATOR IS SQAURE OF SUMS OF COMPARISONS OF VALUES
C OF EQUAL DEPTH FROM END POINTS, E.G., WITH N DATA ELMS,
C X(N)-X(1),X(N-1)-X(2); A(I) IS MULTIPLIED TIMES THE
C DIFFERENCE
      REAL LIST(400),S(400),F(400),RP(50),FN,TSTAT,MU,VAR,
     *NUM,DENOM,MU,VAR,TSUP,SUMWT
      INTEGER LPTR(17),IP(50),NOF,IDIST,HYP,K,KEY(400)
      N=LPTR(2)-1
      CALL MUVAR(LIST,N,MU,VAR)
      DENOM=FLOAT(N-1)*VAR
      DO  5 I=1,N
         KEY(I)=I
 5    CONTINUE
      CALL SHSORT(LIST,KEY,N)
      K=IP(1)
      SUMWT=0.0
      DO 10 I=1,K
         TEMP=RP(1)*(LIST(N-I+1)-LIST(I))
         WRITE(9,210) RP(1),LIST(N-I+1),LIST(I),TEMP
 210     FORMAT(1X,'RP,LISTHI,LISTLO,TEMP ',4F10.4)
         SUMWT=SUMWT+TEMP
 10   CONTINUE
      NUM=SUMWT*SUMWT
      T3=NUM/DENOM
      WRITE(NOF,101) N,MU,VAR,T3,NUM,DENOM
 101  FORMAT(10X,'**** SHAPIRO-WILK TEST FOR NORMALITY ****'
     */1X,'WITH A SAMPLE SIZE OF',I5/1X,'WITH COMPUTED ',
     *'SAMPLE MEAN=',F10.4,', AND VAR=',F10.4/1X,
     *'THE TEST STATISTIC=',F10.4/1X,
     *'FROM NUMERATOR',F10.4,', AND DENOMINATOR=',F10.3)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE SMIRN2 (LIST,LPTR,TREAT,IP,RP,NOF)
C   THE SUBROUTINE TAKES ELEMENTS OF TWO VARIABLES,SORTS
C THEM, AND COMPAIRS THEIR EMPIRICAL CDF'S (FOUND BY
C DIVIDING THE RANK MINUS BY THE TOTAL NUMBER OF ELEMENTS)
C I,M REFER TO FIRST VAR X; J,N REFER TO SECOND VAR Y
C K IS COUNTER FOR DIFFERENCE VECTOR S, L=1 MEANS LAST
C PASS IP(1) IS THE INDEX OF HYPOTHESIS TYPE: 1=HO: NO DIF
C 2=HO:F(X)<=G(Y),  3=HO:F(X)=>G(Y)
      REAL   EPSILN,FN,FM,X(100),Y(100),DIF,ADIF,S(200),TMIN,
     *TMAX,ATMAX,ATMIN,TSUP,PDIF,LIST(400),RP(50),TSTAT
      INTEGER   I,J,K,L,M,N,KEYI(100),KEYJ(100),JJ,IP(50),
     *      LPTR(17),NOF,HYP,II,TREAT(17)
      CALL FINDS(LIST,LPTR,S,K)
      TMIN=2.0
      TMAX=-2.0
      HYP= IP(1)
      DO 35 I=1,K
         IF (S(I).GT.TMAX) TMAX=S(I)
         IF (S(I).LT.TMIN) TMIN=S(I)
 35   CONTINUE
      ATMIN=ABS(TMIN)
      ATMAX=ABS(TMAX)
      TSUP=ATMIN
```

```
      IF (ATMAX.GT.TSUP) TSUP=ATMAX
      WRITE (NOF,102) TREAT(1),TREAT(2),TMIN,TMAX,TSUP
 102  FORMAT(10X,'***** KOLMOLGOROV TEST ******'/1X,
     *' WITH DATA SET VAR #',I3,' AS VARIABLE X '/6X,
     *' AND VAR #',I3,' AS VARIABLE Y'/1X,
     *' THE MAX DIFFERENCE F(X)<G(Y):',F10.4/1X,
     *' THE MAX DIFFERENCE F(X)>G(Y):',F10.4/1X,
     *' THE OVERALL GREATEST DIFFERENCE:',F10.4)
      IF(HYP.EQ.1) TSTAT=TSUP
      IF(HYP.EQ.2) TSTAT=TMAX
      IF(HYP.EQ.3) TSTAT=TMIN
      WRITE(NOF,103) TSTAT
 103  FORMAT(1X/1X,'TEST STATISTIC:',F10.4)
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE CRVONM(LIST,LPTR,TREAT,IP,RP,NOF)
C       THIS SUBROUTINE FINDS THE TEST STATISTIC T2 OF THE
C  CRAMER-VON MISES TEST, AND DOES LINEAR INTERPOLATION
C  TO ARRIVE AT THE ALPHA HAT, ALPHAT.
      REAL   EPSILN,FN,FM,S(400),SSUM,LSIG,ALPHAT,ALPHYP,
     *   T2,ATMAX,ATMIN,TSUP,PDIF,LIST(400),RP(50),CONST
      INTEGER J,K,L,M,N,KEYI(100),KEYJ(100),II,JJ,IP(50),
     *       LPTR(17),NOF,I,TREAT(17)
      CALL FINDS(LIST,LPTR,S,K)
      M=LPTR(2)-1
      N=LPTR(3)-LPTR(2)
      CONST=FLOAT(M*N)/FLOAT((M+N)*(M+N))
      ALPHYP=1.0-RP(1)
      SSUM=0.0
      DO 35 I=1,K
         SSUM=SSUM+(S(I)*S(I))
 35   CONTINUE
      T2=CONST*SSUM
      WRITE(9,121) M,N,K,CONST,SSUM,T2
 121  FORMAT(1X,'M,N,K,CONST,SSUM,T2:',3I4,3F10.4)
      CALL CRVALF(T2,ALPHAT)
      WRITE (NOF,102) T2,ALPHAT,ALPHYP
 102  FORMAT(10X,'***** CRAMER-VON MISES TEST ******'/1X,
     *' THE TEST STATISTIC T2:',F10.4/1X,
     *' GIVES THE ALPHA HAT OF:',F10.4/1X,
     *' COMPARED TO THE HYPOTHESISED ALPHA:',F10.4/1X,
     *' NULL HYP H0: F(X)=G(X) FOR ALL X',/1X,
     *' ALT  HYP H1: F(X)<>G(X) FOR AT LEAST ONE X')
      IF (ALPHYP.LE.ALPHAT) WRITE (NOF,111)
      IF (ALPHYP.GT.ALPHAT) WRITE (NOF,112)
 111  FORMAT(1X,' DO NOT REJECT THE NULL HYPOTHESIS')
 112  FORMAT(1X,' REJECT THE NULL HYPOTHESIS')
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE CRVALF(T2,ALPHAT)
      REAL W(14),ALF(14),ALPHAT,PART,T2,DE,NU,LSIG
      W(1)=0.0
      W(2)=0.04
      W(3)=0.062
      W(4)=0.079
      W(5)=0.097
      W(6)=0.119
      W(7)=0.147
      W(8)=0.184
      W(9)=0.241
      W(10)=0.347
      W(11)=0.461
      W(12)=0.743
      W(13)=1.168
      W(14)=99
```

```
      ALF(1)=0.0
      ALF(2)=0.1
      ALF(3)=0.2
      ALF(4)=0.3
      ALF(5)=0.4
      ALF(6)=0.5
      ALF(7)=0.6
      ALF(8)=0.7
      ALF(9)=0.8
      ALF(10)=0.9
      ALF(11)=0.95
      ALF(12)=0.99
      ALF(13)=0.9999
      ALF(14)=1.0
      I=0
 10   I=I+1
      IF ((T2.GT.W(I)).AND.(I.LE.13)) GO TO 10
      IM1=I-1
      NU=T2-W(IM1)
      DE=W(I)-W(IM1)
      PART=NU/DE
      LSIG=ALF(IM1)+PART*(ALF(I)-ALF(IM1))
      ALPHAT=1.0-LSIG
      WRITE (9,131) I,PART,T2,ALF(IM1),ALF(I),ALPHAT,
     * W(IM1),W(I),NU,DE
 131  FORMAT(1X,'I,PART,T2,ALF-ALF,ALPHAT',I3,5F10.4/1X,
     *' W(IM1),W(I),NU,DE',4F10.4)
      RETURN
      END
C         SUBROUTINE UTEST
C-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/
C         PURPOSE
C           TEST WHETHER TWO INDEPENDENT GROUPS ARE FROM SAME
C           POPULATION BY MEANS OF MANN-WHITNEY U-TEST
C
C         USAGE
C              CALL UTEST(A,R,N1,N2,U,Z,IER)
C
C         DESCRIPTION OF PARAMETERS
C A    - INPUT VECTOR OF CASES CONSISTING OF TWO INDEPENDENT
C        GROUPS. SMALLER GROUP PRECEDES LARGER GROUP. LENGTH
C        IS N1+N2.
C R    - OUTPUT VECTOR OF RANKS. SMALLEST VALUE IS RANKED 1.
C        LARGEST IS RANKED N. TIES ARE ASSIGNED AVERAGE OF TIE
C        RANKS. LENGTH IS N1+N2.
C N1   - NUMBER OF CASES IN SMALLER GROUP.
C N2   - NUMBER OF CASES IN LARGER GROUP.
C U    - STATISTIC USED TO TEST HOMOGENEITY OF THE TWO
C                 GROUPS (OUTPUT).
C Z    - MEASURE OF SIGNIFICANCE OF U IN TERMS OF NORMAL
C           DISTRIBUTION (OUTPUT).
C IER- 0, IF NO ERROR.
C    - 1, IF ALL VALUES OF ONE GROUP ARE TIED.
C
C         REMARKS
C              Z IS SET TO ZERO IF N2 IS LESS THAN 20
C
C         SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C              RANK
C              TIE
C
C  METHOD
C  DESCRIBED IN S. SIEGEL, 'NONPARAMETRIC STATISTICS FOR THE
C  BEHAVIORAL SCIENCES', MCGRAW-HILL, NEW YORK, 1956,
C       CHAPTER 6
C
C     ..............................................................
```

```
C
      SUBROUTINE UTEST(A,R,N1,N2,U,Z,IER)
      DIMENSION A(1),R(1)
C
C RANK SCORES FROM BOTH GROUP TOGETHER IN ASCENDING ORDER, A
C   ASSIGN TIED OBSERVATIONS AVERAGE OF TIED RANKS
C
      N=N1+N2
      CALL RANK(A,R,N)
      Z=0.0
C
C        SUM RANKS IN LARGER GROUP
C
      R2=0.0
      NP=N1+1
      DO 10 I=NP,N
   10 R2=R2+R(I)
C
C        CALCULATE U
C
      FNX= N1*N2
      FN=N
      FN2= N2
      UP=FNX+FN2*((FN2+1.0)/2.0)-R2
      U=FNX-UP
      IF(UP-U) 20,30,30
   20 U=UP
C
C        TEST FOR N2 LESS THAN 20
C
   30 IF(N2-20) 80,40,40
C
C        COMPUTE STANDARD DEVIATION
C
   40 KT=1
      CALL TIE(R,N,KT,TS)
      IF(TS) 50,60,50
   50 IF (TS-(FN*FN*FN-FN)/12)52,51,52
   51 IER=1
      GO TO 80
   52 S=SQRT((FNX/(FN*(FN-1.0)))*(((FN*FN*FN-FN)/12.0)-TS))
      GO TO 70
   60 S=SQRT(FNX*(FN+1.0)/12.0)
C
C        COMPUTE Z
C
   70 Z=(U-FNX*0.5)/S
   80 RETURN
      END
C     ..................................................
C
      SUBROUTINE RANK(A,R,N)
      DIMENSION A(1),R(1)
C        INITIALIZATION
      DO 10 I=1,N
   10 R(I)=0.0
C        FIND RANK OF DATA
      DO 100 I=1,N
C        TEST WHETHER DATA POINT IS ALREADY RANKED
      IF(R(I)) 20, 20, 100
C        DATA POINT TO BE RANKED
   20 SMALL=0.0
      EQUAL=0.0
      X=A(I)
      DO 50 J=1,N
      IF(A(J)-X) 30, 40, 50
C        COUNT NUMBER OF DATA POINTS WHICH ARE SMALLER
```

```
   30 SMALL=SMALL+1.0
      GO TO 50
C        COUNT NUMBER OF DATA POINTS WHICH ARE EQUAL
   40 EQUAL=EQUAL+1.0
      R(J)=-1.0
   50 CONTINUE
C        TEST FOR TIE
      IF(EQUAL-1.0) 60, 60, 70
C
C        STORE RANK OF DATA POINT WHERE NO TIE
C
   60 R(I)=SMALL+1.0
      GO TO 100
C        CALCULATE RANK OF TIED DATA POINTS
   70 P=SMALL + (EQUAL + 1.0)*0.5
      DO 90 J=I,N
      IF(R(J)+1.0) 90, 80, 90
   80 R(J)=P
   90 CONTINUE
  100 CONTINUE
      RETURN
      END
C     .......................................................
      SUBROUTINE TIE(R,N,KT,T)
      DIMENSION R(1)
C        INITIALIZATION
      T=0.0
      Y=0.0
    5 X=1.0E38
      IND=0
C
C        FIND NEXT LARGEST RANK
C
      DO 30 I=1,N
      IF(R(I)-Y) 30,30,10
   10 IF(R(I)-X) 20,30,30
   20 X=R(I)
      IND=IND+1
   30 CONTINUE
C        IF ALL RANKS HAVE BEEN TESTED, RETURN
      IF(IND) 90,90,40
   40 Y=X
      CT=0.0
C        COUNT TIES
      DO 60 I=1,N
      IF(R(I)-X) 60,50,60
   50 CT=CT+1.0
   60 CONTINUE
C        CALCULATE CORRECTION FACTOR
      IF(CT) 70,5,70
   70 IF(KT-1) 75,80,75
   75 T=T+CT*(CT-1.)/2.0
      GO TO 5
   80 T=T+(CT*CT*CT-CT)/12.0
      GO TO 5
   90 RETURN
      END
```

APPENDIX G

PASCAL SOURCE CODE: LOCHINVAR

```
PROGRAM LOCHI (INPUT,OUTPUT,QIN,COUT,A,B,C,D,E,F,CP);
CONST
        TOT_DATA=40C;
        TOT_VARS=16;
        L_NAMES=50;
        L_R_PAR=50;
        L_I_PAR=50;
        L_E_B=50;
TYPE
        INT=INTEGER;
        OUTFILE=11..16;OUTFSET=SET OF OUTFILE;
        T_W_C=1..50;T_W_CEL=SET OF T_W_C;
        HELDER=1..255;HELDEST=SET OF HELDER;
        TESTER=1..25;TEST_N=SET OF TESTER;
        VECTOR= ARRAY(.1..TOT_DATA.) OF REAL;
        VECTORI= ARRAY(.0..TOT_VARS.) OF INTEGER;
        STRING= PACKED ARRAY(.1..L_NAMES.) OF CHAR;
        MATR= ARRAY (.1..15.) OF STRING;
        R_PAR= ARRAY(.0..L_R_PAR.) OF REAL;
        I_PAR= ARRAY(.0..L_I_PAR.) OF INTEGER;
        VEC_E_B=ARRAY (.0..L_E_B.) OF INTEGER;
VAR
        CELL_T:T_W_CEL;
        STATE:TEST_N;
        V_NAME:MATR;
        CELIST,I_P:I_PAR;
        PART_VAL,R_F:R_PAR;
        E_B:VEC_E_B;
        F_NAME,H_A,H_B,H_C:STRING;
        SPEC,OLD_F,CELL_CT,OKAY,EQUAL_S,RIGHT_F,ALL_V_IN,
                NOT_ECL,CH_MIND,GOOD_VAR:BOOLEAN;
        OP,C,D,E,F,CIN,COUT,A,B:TEXT;
        V_T,LIST_PTR,N_ELMS:VECTORI;
        LIST:VECTOR;
        X,Y,YY,MU,VARIANCE,XX:REAL;
        N_EQ_S,CH_FILE,CH_ONE,I,J,K,N_VARS,LIST_C,N_I_P,N_R_P,
          CELLS,N_IN_F,N_OUT_F,N_C_T,MAX_R,MAX_C,N_ROW,N_COL,DICTYP,
          NIF,FILE_TYP,V_OR_C,N_TEST,TYP_CEL,IP1,DICOP,N_TO_RD,
          IN_V_T,S_HI_LST,N_V_T,N_E_B,VAR_C:INTEGER;
        OUTF,EMPTY,Y_OR_N,MORE,CH_1,H,TYPDIC:CHAR;
        OUTSET:OUTFSET;
(*-----------------------------------------------------------*)
(* OUTPUT FOR OPTIONS FILE IS WRITTEN IN SIX PLACES: IN AND
        OUT FILE # IN FILE_IO, THE TEST NUMBER IN W_NTEST,
        IF CELL COUNTS ARE USED--IN W_CELLS,
        VARIABLES INCLUDED IN TEST IN WHICH_VAR,
        EXCLUDED BLOCKS IN EX_BLOC, AND OTHER PARAMETER IN W_PARAM*)
(*----------------------------------------------------------*)
```

```
(*  VARIABLES:
  CELL_CT        BOOLEAN: TRUE IF USING CELL COUNTS FOR TEST
  CELL_CTR       BOOLEAN: TRUE IF USING CELL COUNTS FOR TEST
  CELIST         INTEGER VECTOR: CONTAINS CELL COUNTS FOR ANY TEST
                  FIXED ROW TOTALS FOR R*C TEST
  EQUAL_S        BOOLEAN: TRUE IF EACH VARIABLE HAS SAME # ELEMENTS
  H_A,H_C        CHARACTER STRING: STATEMENTS OF HYPOTHESES
  I_P            INTEGER VECTOR: INTEGER PARAMETERS
  IN_V_T         INTEGER VECTOR: #ELEMS IN EACH VARIABLE
  F_NAME         CHARACTER STRING: NAME OF DATA SET
  FILE_TYP
  LIST           REAL VECTOR: ALL DATA ELEMENTS, VARIABLE BY VARIABLE
  N_IN_F         INTEGER: NUMBER OF THE INPUT FILE
  N_OUT_F        INTEGER: NUMBER OF THE OUTPUT FILE
  N_VARS         INTEGER: NUMBER OF VARIABLES IN THE DATA SET
  N_C_T
  N_I_P          INTEGER: NUMBER OF INTEGER PARAMETERS
  N_R_P          INTEGER: NUMBER OF REAL PARAMETERS
  N_ROW          INTEGER: IN CELL COUNTS, # ROWS IN CELL MATRIX
  N_COL          INTEGER: IN CELL COUNTS, # COLS IN CELL MATRIX
  N_TEST         INTEGER: # OF CHOSEN TEST
  N_ELMS         INTEGER VECTOR: N_ELMS(.0.)= # VARS IN DATA SET/
                  N_ELMS(.I.)= # ELEMENTS IN ITH VARIABLE
  N_EQ_S         INTEGER: 0 IF VARS DON'T HAVE EQUAL # ELEMENTS/
                   ELSE = # ELEMENTS PER VARIABLE
  PART_VAL       REAL VECTOR: PARTITION VALUES FOR CELL COUNTS
                                                                  *)
PROCEDURE RST;(*ALLOWS NEW ENTRIES FROM THE TERMINAL*)
BEGIN
      TERMIN(CIN);
END;
(*-------------------------------------------------------------------*)
FUNCTION YES: BOOLEAN;
BEGIN
      REPEAT  RST;
           WRITELN(QOUT,' ENTER Y OR N');
           READLN(QIN,Y_OR_N);I:=ORD(Y_OR_N);
           IF (I > 200) THEN WRITELN(QOUT,
         ' YOU USED ALTERNATE <APL> CHARACTERS. TRY AGAIN');
      UNTIL (I IN (.149,168.));
      YES:=(I=168);
END;
(*-------------------------------------------------------------------*)
PROCEDURE W_NTEST(N_TEST:INT);
BEGIN
      CELLS:=0;
      (*****ADD IN THE TOTAL NUMBERS OF NTEST WITH CELLS****)
      IF (N_TEST IN CELL_T) THEN CELLS:=1;
```

```
      WRITELN(CP,N_TEST:5,CELLS:5);
END; (*W_TEST*)
(*----------------------------------------------------------------------*)
PROCEDURE W_CELLS(TYP_CEL,IP1,N_COL,N_TO_RD:INT;PART_VAL:R_PAR;
                  CELIST:I_PAR);
VAR I,J:INT;
(*TYP_CEL:1-PARTITION BINOM 2-LIST BINOM,3-PARTITION,
          4-FIXED ROW/COL TOTALS,5-DIRECT ENTRY OF CELL TOTALS
          60'S-MCNEMAR:61-CELL COUNT&NAME IN TEST CP FILE,
               62-CELL COUNT IN TEST OP FILE,NAME IN DATA FILE
               63-CATA PARTITIONED, NAME IN DATA FILE,
          70'S-COCHRAN:71-CELL COUNT & NAME IN TEST OP FILE,
               72-CELL COUNT IN TEST OP FILE,NAME IN DATA FILE
               73-CATA PARTITICNED, NAME IN DATA FILE *)
(*IP1: FOR 1,2:SUCCESS HIGH/ON LIST:=1//FCR 3-5: N_ROW*)
BEGIN
      WRITELN(CP,TYP_CEL:5,IP1:5,N_CCL:5,N_TC_RD:5);
      IF (TYP_CEL IN (.1..3,63,73.)) THEN BEGIN
         IF (N_TO_RD>0) THEN
           FCR I:=1 TC N_TO_RD DO     BEGIN
              WRITE(OP,PART_VAL(.I.):10:4);
            IF ((I=N_TO_RD) CR (0=(I MOD 8))) THEN WRITELN(OP);END;
      END; (*IF TYP_CEL 1,2,3,63,73 *)
      IF (TYP_CEL IN (.4,5,61,62,71, 72.)) THEN BEGIN
         IF (N_TO_RD>0) THEN
           FOR I:=1 TO N_TO_RD DO    BEGIN
             WRITE(OP,CELIST(.I.):5);
             IF ((I=N_TO_RD) OR (0=(I MOD 16))) THEN WRITELN(OP);END;
      END; (*IF TYP_CEL 1,2,3,61,62, 71,72*)
END; (*W_CELLS*)
(*------------------------------------------------------------*)
PROCEDURE SETT_CELL(N_EQ_S,MAX_R,MAX_C:INT;VAR N_ROW,N_COL:INT);
VAR    JJ,P_F,K,HOLD,SIZE_ROW,SIZE_COL,IM1,JM1,I,J:INT;
(* ROW REFERS TO FIRST VAR OF PAIR, COL REFERS TO SECOND OF PAIR*)
BEGIN         CELIST(.1.):=0;PART_VAL(.1.):=0;
     IF (MAX_R=2) THEN N_ROW:=2;IF (MAX_C=3) THEN N_COL:=2;
     IF (MAX_R>2) CR (MAX_C>2) THEN
        REPEAT RST;
           WRITELN(QOUT,'ENTER # ROWS= # DIVISIONS IN 1ST VARIABLE');
         READLN(CIN,N_ROW);
         WRITELN(QCUT,'ENTER # COLUMNS= # DIVISIONS IN 2D VAR');  RST;
         READLN(QIN,N_COL);WRITELN(QOUT,N_ROW:5,N_COL:5,'CORRECT?');
        UNTIL YES;
   WRITELN(QCLT,'YOU MAY ENTER PARTITION VALUES TC DIVIDE THE VALUES',
     'OF THE FIRST VARIABLE INTO ROWS, THE SECOND VAR INTO COLUMNS');
   WRITELN(QCLT,' OR YOU MAY SPECIFY FIXED RCW ANC COLUMN TOTALS');
    REPEAT RST; WRITELN(QOUT,'ENTER P FOR PARTITON OR F FOR FIXED');
      READLN(CIN,H);P_F:=CRD(H)     UNTIL (P_F IN (.134,151.));
```

```
IF (P_F=151) THEN   BEGIN(*// PARTITICN//*)
  REPEAT  (*START ROW PARTITION *)
    FOR J:=2 TO N_ROW DO BEGIN JM1:=J-1;
        WRITELN(QOUT,'ENTER PARTITION VALUE THAT SEPARATE S');
        WRITELN(QOUT,'ROW',JM1:5,' AND ROW',J:5);
        RST;READLN(QIN,PART_VAL(.JM1.)); END; (*FOR J*)
    FOR J:=1 TO (N_ROW-1) DO
         WRITELN(QCUT,J:5,PART.VAL(.J.):10:3,'CORRECT?')
  UNTIL(YES); (*END ROW PARTITIONS *)
  REPEAT   (* START CCL PARTITIONS *)
     FOR I:=2 TO N_CCL DO BEGIN IM1:=I-1;
         WRITELN(QOUT,'ENTER PARTITION VALUE THAT SEPARATES');
         WRITELN(QCUT,'COL',IM1:5,' AND CCL',I:5);
         RST;READLN(QIN,PART_VAL(.(IM1+N_RCW-1).)); END;(*FOR I*)
     FOR I:=1 TO (N_CCL-1) DO
           WRITELN(QOUT,PART_VAL(.(I+N_ROW-1).):10:3,'CORRECT?')
  UNTIL(YES);  (*END COL PARTITIONS*)
  W_CELLS(3,N_ROW,N_CCL,(N_RCW+N_COL-2),PART_VAL,CELIST);
END;(*IF P_F=151*)(*// END PARTITION //*)
IF (P_F=134) THEN BEGIN  (*// FIXED ROW/CCL TOTALS //*)
  SIZE_RCW:=N_EQ_S;SIZE_CCL:=N_EQ_S;J:=1; I:=1;
  WRITELN(QOUT,'THERE ARE A TOTAL OF',N_EQ_S:5,' PAIRS',
        ' WHICH MAY BE DIVIDED INTO ROWS ANC COLUMNS');
 REPEAT   (* FIXED ROW TOTALS *)
  REPEAT
     WRITELN(COUT,SIZE_RCW:5,' PAIRS LEFT FOR ROW ASSIGNMENT');
     RST; WRITELN(QOUT,'ENTER FIXED TCTAL FOR ROW',I:5);
     READLN(QIN,HOLD); IF (HOLD<=SIZE_ROW) THEN BEGIN
          CELIST(.I.):=HOLD;SIZE_ROW:=SIZE_ROW-HOLD; I:=I+1;END;
  UNTIL (SIZE_ROW<=0);K:=1; I:=I-1;
  IF (I<>N_ROW) THEN   BEGIN   REPEAT
        WRITELN(QOUT,'# VALS FOR ROW TOTALS,',I:5,' NOT EQUAL TO',
          ' PREVIOUSLY ENTERED #ROWS',N_RCW:5,' ENTER N TO CHANGE',
          ' # ROWS OR V TO CHANGE VALUES');
        RST;READLN(QIN,H);K:=ORD(H) UNTIL (K IN (.149,165.));
     IF(K=149) THEN N_ROW:=I;
     IF(K=165) THEN BEGIN SIZE_RCW:=N_EQ_S;I:=1; END;(*IF K=165*)
  END; (*IF I<>N_ROW*)
 UNTIL (I=N_ROW);(*UNTIL N_ROW = #ROW TOTALS*)
 REPEAT  (*FIXED CCL TOTALS*)
  REPEAT  JJ:=J+N_ROW;
     WRITELN(COUT,SIZE_CCL:5,' PAIRS LEFT FOR COL ASSIGNMENT');
     RST; WRITELN(QOUT,'ENTER FIXED TCTAL FCR COL',J:5);
     READLN(QIN,HOLD); IF (HOLD<=SIZE_COL) THEN    BEGIN
          CELIST(.(JJ).):=HOLD;SIZE_CCL:=SIZE_COL-HOLD;
          J:=J+1;END;(*IF HOLD<=SIZE_COL*)
  UNTIL (SIZE_COL<=0);J:=J-1;
  IF (J<>N_COL) THEN   BEGIN   REPEAT
```

```
         WRITELN(QOUT,'# VALS FOR COL TOTALS,',J:5,' NOT EQUAL TO',
           ' PREVIOUSLY ENTERED #COLS',N_COL:5,' ENTER N TO CHANGE',
           ' # COLS OR V TO CHANGE VALUES');
         RST;READLN(QIN,H);K:=ORD(H) UNTIL (K IN (.149,165.));
       IF(K=149) THEN N_COL:=J;
       IF(K=165) THEN BEGIN SIZE_COL:=N_EQ_S;J:=1; END;(*IF K=165*)
     END; (* J<>N_COL *)
   UNTIL (J=N_COL);(*N_COL = #COL TOTALS*)
     W_CELLS(4,N_ROW,N_COL,(N_ROW+N_COL),PART_VAL,CELIST);
  END; (*IF P_F=134*)  (*// FIXED ROW/COL TOTALS //*)
END; (*SETT_CELL*)
(*-------------------------------------------------------------*)
PROCEDURE DICOT;(* GETS RULE TO DICOTOMIZE DATA SET *)
VAR I:INTEGER;
BEGIN       CELIST(.1.):=0;
  WRITELN(QOUT,'TO PERFORM THIS TEST, THE DATA MUST BE ',
   ' DIVIDED INTO SUCCESSES AND FAILURES:',
   ' EITHER PARTITION (I.E., DIVIDE DATA INTO TWO  ',
   ' SETS, ONE ABOVE, ONE BELOW THE PARTITION VALUE), OR LIST THE ',
   ' VALUES OF ONE SET');
    REPEAT RST;WRITELN(QOUT,' ENTER P FOR PARTITION, L FOR LIST');
       READLN(QIN,TYPDIC);DICTYP:=ORD(TYPDIC)
    UNTIL (DICTYP IN (.147,151.));
   IF (DICTYP=151) THEN        BEGIN
    REPEAT RST; WRITELN(QOUT,' ENTER PARTITION VALUE');
     READLN(QIN,PART_VAL(.1.));
     WRITELN(QOUT,PART_VAL(.1.):9:4,' CORRECT?')
    UNTIL (YES); S_HI_LST:=0;(*IMPLIES FAILURES HIGH: > PARTITION*)
    WRITELN(QOUT,'ARE SUCCESSES THOSE VALUES ABOVE THE PARTITION?');
    IF (YES) THEN S_HI_LST:=1;W_CELLS(1,S_HI_LST,2,1,PART_VAL,CELIST);
   END; (*IF DICTYP=151*)
   IF (DICTYP=147) THEN       BEGIN
     REPEAT WRITELN(QOUT,' ENTER THE NUMBER OF DIFFERENT VALUES IN',
        ' LIST (SMALLER OF TWO POSSIBLE)'); RST; READLN(QIN,DICOP);
        WRITELN(QOUT,DICOP,' CORRECT?')   UNTIL (YES);
     REPEAT
        FOR I:=1 TO DICOP DO   BEGIN  WRITELN(QOUT,' ENTER VALUE',
            I); RST;READLN(QIN,PART_VAL(.I.));    END;
        WRITELN(QOUT,' THESE ARE THE VALUES YOU ENTERED:');
        FOR I:=1 TO DICOP DO WRITE(QOUT,PART_VAL(.I.):10:4);
         WRITELN(QOUT);WRITELN(QOUT,' CORRECT?')
     UNTIL (YES);S_HI_LST:=0;(*IMPLIES FAILURES WERE LISTED*)
     WRITELN(QOUT,'WERE THE LISTED VALUES SUCCESSES?');
     IF (YES) THEN S_HI_LST:=1;
     W_CELLS(2,S_HI_LST,2,DICOP,PART_VAL,CELIST);
   END; (*DICYPT=147*)
END; (*PROCEDURE DICOT*)
(*-------------------------------------------------------------*)
```

```
FUNCTION HYP(H_A,H_B,H_C:STRING;K:INT):INT;
VAR HYPER:CHAR;
BEGIN
   WRITELN(QOUT,' TEST HYPOTHESES ARE EXPLAINED ON CONNOVER P.',K:5);
   WRITELN(QOUT,H_A);WRITELN(QOUT,H_B);WRITELN(QOUT,H_C);
   REPEAT RST;WRITELN(QOUT,' ENTER A,B,OR C');
      READ(QIN,HYPER); I:=ORD(HYPER)-128
   UNTIL I IN (.1,2,3.);
   HYP:=I;
END;
(*------------------------------------------------------------------*)
PROCEDURE W_PARAM(N_I_P,N_R_P:INT;I_P:I_PAR;R_P:R_PAR);
VAR I:INT;
BEGIN
      WRITELN(OP,N_I_P:5);
      IF (N_I_P>0) THEN FOR I:=1 TO N_I_P DO   BEGIN
         WRITE(OP,I_P(.I.):5);
         IF((I=N_I_P) OR (0=(I MOD 16))) THEN WRITELN(OP); END;
      WRITELN(OP,N_R_P:5);
      IF (N_R_P>0) THEN FOR I:=1 TO N_R_P DO   BEGIN
         WRITE(OP,R_P(.I.):10:3);
      IF (I=N_R_P) OR (0=(I MOD 8)) THEN WRITELN(OP); END;
END; (*W_PARAM*)
(*------------------------------------------------------------------*)
PROCEDURE NONE_EX;
VAR I:INT;
BEGIN
      I:=0;
      WRITELN(OP,I:5);
END;(*NONE_EX*)
(*------------------------------------------------------------------*)
FUNCTION P_C:REAL;
VAR PERCENT:REAL;
BEGIN REPEAT RST;
     WRITELN(QOUT,'ENTER NUMBER>0, IF <1, START WITH 0.');
     READ(QIN,PERCENT) UNTIL (PERCENT>=0) AND (PERCENT<=1);
     P_C:=PERCENT;
END;(*P_C*)
(*------------------------------------------------------------------*)
FUNCTION L_SIG:REAL;(*RETURNS A LEVEL OF SIGNIFICANCE*)
BEGIN
     WRITELN(QOUT,' ENTER LEVEL OF SIGNIFICANCE, ALPHA');
     L_SIG:=1.0-(P_C);
END;
(*------------------------------------------------------------------*)
PROCEDURE FILE_IO(VAR NIF,N_IN_F:INTEGER);
VAR J,K,N_OUT_F,L:INTEGER;
BEGIN  REWRITE(OP,'NAME = LOCHI.OP.A,RECFM=F,LRECL=80');L:=0;
```

```
    REPEAT RST;WRITELN(QOUT,' WHICH FILE NAME FOR OUTPUT? U-Z');
      READLN(QIN,OUTF);K:=ORD(OUTF)-153 UNTIL (K IN (.11..16.));
      NIF:=N_IN_F-128;(*NIF IS NUMBER READ BY FORTRAN PROGRAM *)
      IF (NIF=6) THEN NIF:=7;(*TO PROTECT TEST F FROM INTERFERENCE*)
      WRITELN(OP,NIF:5,K:5);
END;
(*-----------------------------------------------------------------*)
PROCEDURE WHICH_VEC(N_VARS,N_C_T:INT);
    (*FINDS WHICH VARIABLES OR TREATMENTS TO USE IN TEST*)
VAR V1,V2,I:INT;
    HELD:HELDSET;
    HOLDING:BOOLEAN;(*TRUE IF HELD IS NOT EMPTY*)
BEGIN
  HOLDING:=FALSE;
  IF (N_VARS>1) AND (N_C_T=1) THEN REPEAT RST;N_V_T:=1;
    WRITELN(QOUT,'OF',N_VARS:3,' VARIABLES IN DATA SET,');
    WRITELN(QOUT,' WHICH VARIABLE FOR THIS TEST?');
     READ(QIN,V_T(.1.)) UNTIL (V_T(.1.)<=N_VARS);
  IF (N_VARS=1) AND (N_C_T=1) THEN V_T(.1.):=1;N_V_T:=1;
  IF (N_VARS>2) AND (N_C_T=2) THEN BEGIN REPEAT RST;N_V_T:=2;
   WRITELN(QOUT,'OF',N_VARS:3,' VARIABLES IN DATA SET,');
   WRITELN(QOUT,' WHICH 2 VARIABLES FOR THIS TEST?');
   READ(QIN,V1,V2) UNTIL (V1<=N_VARS) AND (V2<=N_VARS);
   HELD:=(.V1,V2.);HOLDING:=TRUE;END;
  IF (N_VARS=2) AND (N_C_T=2) THEN BEGIN N_V_T:=2;
      V_T(.1.):=1;V_T(.2.):=2;END;
  IF (N_VARS>2) AND (N_C_T>2) THEN
    BEGIN
      WRITELN(QOUT,'DO YOU WANT ALL ',N_VARS:3,'INCLUDED IN TEST ');
      IF (YES) THEN BEGIN FOR I:=1 TO N_VARS DO V_T(.I.):=I;
              N_V_T:=N_VARS;END
      ELSE
        BEGIN
           HOLDING:=TRUE;HELD:=(.255.);RST; REPEAT
           WRITELN(QOUT,'HOW MANY VARIABLES ARE INCLUDED IN TEST?');
           READ(QIN,N_V_T) UNTIL (N_V_T<=N_VARS);RST;
           WRITELN(QOUT,' ENTER INDICES OF VARIABLES TO BE IN TEST');
           FOR I:=1 TO N_V_T DO
           REPEAT OKAY:=FALSE;RST;
              WRITELN(QOUT,' ENTER INDEX OF INCLUDED VARIABLE',I:3);
              READ(QIN,V1);IF(V1<=N_VARS) AND NOT(V1 IN HELD) THEN
                  BEGIN HELD:=HELD+(.V1.);OKAY:=TRUE; END
           UNTIL OKAY;
        END;
    END;
    J:=0;
    IF HOLDING THEN
      FOR I:=1 TO N_VARS DO
```

```
          IF (I IN HELD) THEN BEGIN
               J:=J+1; V_T(.J.):=I;END;
    WRITELN(OP,N_V_T:5);
    IF (N_V_T>1) TFEN FOR I:=1 TO (N_V_T-1) DO WRITE(OP,V_T(.I.):5);
    IF (N_V_T>C) TFEN WRITELN(OP,V_T(.N_V_T.):5);
END; (*WHICH_VEC*)
(*-----------------------------------------------------------------*)
PROCEDURE EXCLUD_E(N_EQ_S:INT);
VAR  I,J,K,L,FCLD:INT;
     XCLUD:BOCLEAN; (*TRUE IF WANT TO EXCLUDE SOME TREATMENTS*)
     GOOD,BAD,CUPE:VEC_E_B;
     HELD:HELCEST;
BEGIN
   WRITELN(QOUT,' CO YOU WISH TO EXCLUDE ANY BLOCKS FROM TEST');
     XCLUD:=YES; IF NOT XCLUD THEN NONE_EX;
   IF XCLUD THEN
     BEGIN RST;
       I:=0;J:=0;K:=0; (*COUNTERS FOR HOLD,GOCC, BAD VECTORS*)
       HELD:=(.(N_EQ_S+1)..(N_EQ_S+1).);
       WRITELN(COUT,'HCW MANY BLOCKS DO YOU WISH TC EXCLUDE?');
       REPEAT WRITELN(QOUT,' UP TO ',N_EQ_S:3,' BLCCKS');RST;
           READLN(QIN,N_E_B) UNTIL (N_E_B<N_EQ_S);
       IF (N_E_B>0) THEN FOR I:= 1 TO N_E_B DO
        BEGIN
           WRITELN(COUT,'IN ASCENDING ORDER,',
                   ' ENTER INDICES, 1 -',N_EC_S:4);
           REPEAT
              WRITELN(QOUT,' ENTER INDEX FOR EXCLUDED BLOCK NR',I:3);RST;
               READ (QIN,HOLD);CKAY:=FALSE;
               IF(HOLD IN (.1..N_EQ_S.)) AND (NOT(HOLD IN HELD)) THEN
                   BEGIN OKAY:=TRUE;HELD:=HELD+(.HOLD.);END
           UNTIL OKAY;
          END; J:=0;
       FOR I:=1 TO N_EC_S DO
           IF (I IN HELD) THEN BEGIN
                J:=J+1; E_B(.J.):=I;END;
       WRITELN(OP,N_E_B:5);
       IF (N_E_B>0) THEN FOR L:=1 TO N_E_B DO   BEGIN
            WRITE(OF,E_B(.L.):5);
       IF ((L=N_E_B) OR (0=(L MOD 16))) THEN WRITELN(OP);END;
     END;
END;
(*-----------------------------------------------------------------*)
PROCEDURE CEL_CTR(MAX_R,MAX_C:INT;VAR N_ROW,N_COL:INT);
VAR  J,I:INT;
BEGIN           FART_VAL(.1.):=0;
   IF(MAX_R=1) AND (MAX_C=2) THEN REPEAT RST; N_RCW:=1;N_COL:=2;
      WRITELN(COUT,' ENTER NUMBER OF SUCCESS, THEN NUMBER OF FAILURES');
```

```
      READLN(QIN,CELIST(.1.),CELIST(.2.));
      WRITELN(QOUT,CELIST(.1.),CELIST(.2.),'CORRECT?')    UNTIL YES;
  IF  (MAX_R>2) OR  (MAX_C>2) THEN    REPEAT RST;
        WRITELN(QOUT,'ENTER NUMBER OF ROWS AND COLUMNS');
       READLN(QIN,N_ROW,N_COL);
       WRITELN(QOUT,'# ROWS',N_ROW:5,' # COLUMNS',N_COL:5,' CORRECT?');
          UNTIL YES;
      IF (MAX_R=2) THEN N_ROW:=2; IF (MAX_C=2) THEN N_COL:=2;
      IF (MAX_R>1) THEN     BEGIN
          WRITELN(QOUT,' ENTER CELL COUNTS BY ROW');
          REPEAT   FOR I:=1 TO N_ROW DO
                   FOR J:=1 TO N_COL DO  BEGIN RST;
                    WRITELN(QOUT,' ENTER COUNT FOR ROW',I:5,'COL',J:5);
                     READLN(QIN,CELIST(.(J+(I-1)*N_COL).));  END;
                 FOR I:=1 TO N_ROW DO  BEGIN
                   FOR J:=1 TO (N_COL-1) DO
                      WRITE(QOUT,CELIST(.(J+(I-1)*N_COL).):5);
                   WRITELN(QOUT,CELIST(.(I*N_COL).):5); END;
                WRITELN(QOUT,' CORRECT?');
          UNTIL YES;
       END; (* IF MAX_R > 1 *)
     W_CELLS(5,N_ROW,N_COL,(N_ROW*N_COL),PART_VAL,CELIST);
END; (*CEL_CTR*)
(*-----------------------------------------------------------------------*)
PROCEDURE MONCREG(N_VARS,N_EQ_S: INT);
VAR  I,K: INT;
BEGIN
     WHICH_VEC(N_VARS,2);
     EXCLUD_B(N_EQ_S);
     WRITELN(QOUT,'IN REGRESSION, X IS INDEPENDENT VARIABLE, AND ');
     WRITELN(QOUT,'  Y IS THE DEPENDENT VARIABLE, SUCH THAT');
     WRITELN(QOUT,'  E(Y|X)=E(Y)+B*(X-E(X)), B<>0 ');
     WRITELN(QOUT,'IS THE LOWER INDEXED VARIABLE OF THE TWO THE');
     WRITELN(QOUT,'  DEPENDENT VARIABLE?');
     IF YES THEN I_P(.1.):=1 ELSE I_P(.1.):=2;
     WRITELN(QOUT,'DO YOU WANT TO PREDICT VALUES OF Y');
     WRITELN(QOUT,'  BASED ON SOME VALUES OF X?');
     IF NOT(YES) THEN I_P(.2.):= 0  ELSE
         BEGIN
             REPEAT
                 RST; WRITELN(QOUT,'ENTER # VALUES TO BE PREDICTED:');
                 READLN(QIN,I);WRITELN(QOUT,I:5,' CORRECT?');
             UNTIL (YES AND (I > 0));
             I_P(.2.):=I;
             K:=I;
             FOR I:=1 TO K DO
               BEGIN
                 REPEAT
```

```
              RST; WRITELN(QOUT,'ENTER X VALUE #',I:5);
              READLN(QIN,X);WRITELN(QOUT,X:10:3,' CORRECT?');
             UNTIL YES;
             R_P(.I.):=X;
           END; (*FOR K DO *)
        END; (*ELSE*)
   WRITELN(QOUT,'DO YOU WANT TO HAVE A TRACE OF THE REGRESSION?');
   IF YES THEN I_P(.3.):=1 ELSE I_P(.3.):=0;
   K:=I_P(.2.);
   W_PARAM(3,K,I_P,R_P);
END;  (*MONOREG*)
(*---------------------------------------------------*)
PROCEDURE GDFIT_U(VAR R_P:R_PAR);
VAR ALPHA,BETA:REAL;
BEGIN
     WRITELN(QOUT,'THE UNIFORM DISTRIBUTION HAS AS PARAMETERS');
     WRITELN(QOUT,'THE ENDPOINTS OF THE DISTRIBUTION');
     WRITELN(QOUT,' ENTER ALPHA(LOWER ENDPOINT) AND BETA');
     REPEAT RST; WRITELN(QOUT,'ALPHA < BETA');
       READLN(QIN,ALPHA,BETA)  UNTIL (ALPHA<BETA);
     I_P(.2.):=0;R_P(.1.):=ALPHA;R_P(.2.):=BETA;
END; (*GDFIT_U*)
(*---------------------------------------------------*)
PROCEDURE GDFIT_EX(VAR I_P:I_PAR;VAR R_P:R_PAR);
VAR LAMBDA:REAL;EST:BOOLEAN;
BEGIN
     WRITELN(QOUT,'THE EXPONENTIAL DISTRIBUTION HAS PARAMETERS');
     WRITELN(QOUT,'LAMBDA,THE RATE OR SCALE PARAMETER');
     WRITELN(QOUT,'YOU MAY ENTER LAMBDA, OR HAVE THE PROGRAM');
     WRITELN(QOUT,'ESTIMATE BY METHOD OF MOMENTS. THE LATTER');
     WRITELN(QOUT,'METHOD WILL USE THE SAMPLE MEAN AS THE');
     WRITELN(QOUT,'RECIPROCAL OF THE RATE. DO YOU WISH TO');
     WRITELN(QOUT,'ENTER A VALUE OF LAMBDA?');
     IF YES THEN   BEGIN   REPEAT   RST;WRITELN(QOUT,'LAMBDA=');
       READLN(QIN,LAMBDA);I_P(.2.):=0;
       WRITELN(QOUT,'LAMBDA=',LAMBDA:10:3,' CORRECT?')
       UNTIL YES; R_P(.1.):=LAMBDA;       END
     ELSE I_P(.2.):=1;
END; (*GDFIT_EX*)
(*---------------------------------------------------*)
PROCEDURE GDFIT_N(VAR I_P:I_PAR;VAR R_P:R_PAR);
BEGIN
   WRITELN(QOUT,'YOU MAY SPECIFY THE THEORETICAL DISTRIBUTIONS');
   WRITELN(QOUT,'TWO PARAMETERS--MEAN AND VARIANCE--OR HAVE THE');
   WRITELN(QOUT,'PROGRAM ESTIMATE FROM THE SAMPLE BY THE METHOD');
   WRITELN(QOUT,'OF MOMENTS. DO YOU WANT TO SPECIFY PARAMETERS?');
   SPEC:=YES; IF SPEC THEN    BEGIN   REPEAT
     WRITELN(QOUT,'ENTER MEAN AND VARIANCE FOR THEORETICAL DIST');
```

```
      RST;READLN(QIN,MU,VARIANCE);WRITELN(QOUT,'MU',MU:10:3,
            ' VAR',VARIANCE:10:3,' CORRECT?') UNTIL YES;
         R_P(.1.):=MU;R_P(.2.):=VARIANCE;I_P(.2.):=0;
        END; (*IF SPEC*)
      IF NOT SPEC THEN I_P(.2.):=2;
END; (*GDFIT_N*)
(*----------------------------------------------------------------*)
PROCEDURE RC_CONT(N_VARS,N_EQ_S:INT;CELL_CT:BOOLEAN);
VAR I,J:INT;
BEGIN
      IF CELL_CT THEN CEL_CTR(25,25,N_ROW,N_COL);
      IF (NOT CELL_CT) THEN  BEGIN
        SETT_CELL(N_EQ_S,25,25,N_ROW,N_COL);
        WHICH_VEC(N_VARS,2);
        EXCLUD_B(N_EQ_S);   END; (*IF NOT CELL_CT*)
I_P(.1.):=0;R_P(.1.):=L_SIG;
      W_PARAM(0,1,I_P,R_P);
END; (* RC_CONT *)
(*--------------------------------------------------------------------*)
PROCEDURE BINOM(N_VARS:INT;CELL_CT:BOOLEAN);
   (*THIS IS TEST PROCEDURE FOR BINOMIAL TEST*)
VAR THIS_VAR,I,J,K:INT;(*THIS_VAR IS INDEX OF A         *)
BEGIN
   IF CELL_CT THEN CEL_CTR(1,2,N_ROW,N_COL);
   IF NOT CELL_CT THEN BEGIN
   DICOT; WHICH_VEC(N_VARS,1);NONE_EX; END; (*IF NOT CELL_CT*)
   N_R_P:=2;N_I_P:=1;
   WRITELN(QOUT,' ENTER HYPOTHIZED P*,E.G.,0.8');
      R_P(.1.):=P_C;
   H_A:=' A:NULL HYPOTHESIS OF FORM:  P=P*                       ';
   H_B:=' B:NULL H:  P<=P*                                        ';
   H_C:=' C:NULL H:  P>=P*                                        ';
   I_P(.1.):=HYP(H_A,H_B,H_C,96);
   R_P(.2.):=L_SIG;
   W_PARAM(1,2,I_P,R_P);
 END;(*BINOM*)
(*--------------------------------------------------------------------*)
PROCEDURE COXSTU(N_VARS,N_EQ_S:INT;EQUAL_S:BOOLEAN);
   (*THIS IS TEST PROCEDURE FOR COX-STUART TEST FOR TREND*)
VAR THIS_VAR,TWO,I,J,K,INDEP,DEP:INT;
BEGIN
I_P(.2.):=1;N_I_P:=2;
IF (N_VARS>1) AND EQUAL_S THEN BEGIN
     WRITELN(QOUT,' YOU HAVE TWO OPTIONS FOR THE COX-STUART TEST:');
     WRITELN(QOUT,'1. CHOOSE ONE VARIABLE, TEST FOR TREND IN THE ',
           ' VALUES IN THEIR CURRENT ORDER');
     WRITELN(QOUT,'2. CHOOSE TWO VARIABLES, TEST FOR POSITIVE OR ',
           ' NEGATIVE CORRELATION, HAVING THE RANKS OF ONE VARIABLE ',
```

```
          ' (INDEPENDENT) ORDER THE VALUES OF THE CTHER (DEPENDENT) ',
          ' VARIABLE FCR THE TEST FOR TRENC.');
    REPEAT RST; WRITELN(QOUT,' ENTER CPTICN (1 OR 2)');
          REACLN(QIN,I)   UNTIL (I IN (.1..2.)); I_P(.2.):=I;
    IF (I_P(.2.)=2) THEN BEGIN
       REPEAT
          REPEAT   RST;
               WRITELN(QOUT,' CF',N_VARS:5,' VARIAELES, WHICH IS ',
                  ' DEPENDENT (WILL HAVE ITS ORDER REARRANGED)');
               REACLN(CIN,DEP) UNTIL (DEP<=N_VARS);
          REPEAT  RST;
             WRITELN(QOUT,' WHICH VARIABLE IS INDEPENDENT?');
             READLN(QIN,INDEP) UNTIL (INDEP<=N_VARS) AND (INDEP<>DEP);
          RST; WRITELN(QOUT,' INDEX OF DEPENDENT:',DEP:5,
                ' INDEPENDENT:',INDEP:5,'CORRECT?')
      UNTIL (YES);I_P(.3.):=DEP;I_P(.4.):=INCEP;N_I_P:=4;
          (* THIS IN LIEU OF WHICH_VAR *)
      I:=DEP;J:=INDEP;TWC:=2;
      IF(DEP>INDEP) THEN BEGIN I:=J; J:=DEP;END;(*IF DEP>INDEP*)
      WRITELN(OP,TWO:5);WRITELN(OP,I:5,J:5);
    END; (*IF IP(.2.)=2*)
END; (*IF N_VARS>1 AND E_SIZE*)
   IF (I_P(.2.)=1) THEN  WHICH_VEC(N_VARS,1);
   IF EQUAL_S THEN EXCLUD_B(N_EQ_S) ELSE NONE_EX;
   N_R_P:=1;
   H_A:=' A: H0: NO TREND H1:TREND OF INCREASE OR DECREASE ';
   H_B:=' B: H0: CECREASE OR NO TREND H1: TREND OF INCREASE';
   H_C:=' C: H0: INCREASE OR NO TREND H1: TREND OF DECREASE';
   I_P(.1.):=HYP(H_A,H_B,H_C,134);
   R_P(.1.):=L_SIG;
   W_PARAM(N_I_P,1,I_P,R_P);
 END;(*COX STLART*)
(*-----------------------------------------------------------------------*)
PROCEDURE QUANTILE(N_VARS:INT;CELL_CT:BOOLEAN);
   (*THIS IS TEST PROCEDURE FOR QUANTILE TEST*)
VAR THIS_VAR,I,J,K:INT;(*THIS_VAR IS INDEX OF A           *)
BEGIN
   N_R_P:=2;N_I_P:=1;
   WRITELN(QCUT,' ENTER QUANTILE (I.E.,P* OF Q(P)),E.G.,0.8');
     R_P(.1.):=P_C;
   RST; WRITELN(QCUT,' ENTER HYPOTHESIZED X* FOR Q(P)');
      READ(QIN,PART_VAL(.1.));CELIST(.1.):=1;
      W_CELLS(1,0,2,1,PART_VAL,CELIST);
   WHICH_VEC(N_VARS,1);
   NCNE_EX;(*NC BLOCKS EXCLUDED*)
   H_A:=' A:NULL HYPOTHESIS OF FORM: P(X<X*)=P*                 ';
   H_B:=' B:NULL H: P(X<X*)<=P*                                 ';
   H_C:=' C:NULL H: P(X<=X*)>=P*                                ';
```

```
   I_P(.1.):=HYP(H_A,H_B,H_C,106);
   R_P(.2.):=L_SIG;
   W_PARAM(1,2,I_P,R_P);
 END;(*QUANTILE*)
(*----------------------------------------------------------*)
PROCEDURE MANWHT(NVARS:INT);
VAR I,J,K,N_E_B:INT;
BEGIN
      WHICH_VEC(N_VARS,2);
      R_P(.1.):=L_SIG;I_P(.1.):=0;N_E_B:=0;
      WRITELN(OP,N_E_B:5);
      W_PARAM(0,1,I_P,R_P);
END;
(*----------------------------------------------------------*)
PROCEDURE SIGNTEST(NVARS,N_EQ_S:INT);
VAR I,J,K:INT;
BEGIN
      WHICH_VEC(N_VARS,2);
      EXCLUD_B(N_EQ_S);
   H_A:=' A:NULL HYP HO: P(+)=P(-),ALT HYP H1: P(+)<>P(-)  ';
   H_B:=' B:NULL HYP HO: P(+)<=P(-),ALT HYP H1: P(+)>P(-)  ';
   H_C:=' C:NULL HYP HO: P(+)>=P(-),ALT HYP H1: P(+)<P(-)  ';
   I_P(.1.):=HYP(H_A,H_B,H_C,123);
      R_P(.1.):=L_SIG;
      W_PARAM(1,1,I_P,R_P);
END;(*SIGNTEST*)
(*----------------------------------------------------------*)
PROCEDURE FRIEDMAN(N_VARS,N_EQ_S:INT);
VAR I,J,K:INT;XX:REAL;
BEGIN
  WHICH_VEC(N_VARS,N_VARS);
  EXCLUD_B(N_EQ_S);
  WRITELN(QOUT,'   THIS WILL ALLOW THE FRIEDMAN OR DURBAN TEST');
  WRITELN(QOUT,'<DURBAN IS INCOMPLETE BLOCK DESIGN: NEEDS FILLER>');
  WRITELN(QOUT,'   DO YOU WANT TO PERFORM THE FRIEDMAN TEST?');
  IF YES THEN BEGIN
      R_P(.1.):=L_SIG;I_P(.1.):=0;
      W_PARAM(1,1,I_P,R_P); END  (*IF*)
    ELSE  BEGIN
      REPEAT    RST;
         WRITELN(COUT,'ENTER THE FILLER VALUE');
         READLN(QIN,XX);WRITELN(QOUT,XX:10:3,' CORRECT?');
      UNTIL YES;
      REPEAT    RST;
         WRITELN(COUT,'ENTER K=# TREATS/BLOCK');
         WRITELN(COUT,'AND R=# BLOCKS/TREAT');
         READLN(QIN,I,J);WRITELN(QOUT,I:5,J:5,' CORRECT?');
      UNTIL YES;
```

```
        I_P(.2.):=I;I_P(.3.):=J;
        R_P(.2.):=XX;I_P(.1.):=1;R_P(.1.):=L_SIG;
        W_PARAM(3,2,I_P,R_P);
     END;  (*ELSE*)
END;(*FRIEDMAN*)
(*-----------------------------------------------------------------*)
PROCEDURE MEC_AG(N_VARS:INT;VAR I_P:I_PAR;VAR N_I_P:INT);
VAR KK,I:INT;
BEGIN
     N_I_P:=0;
     FOR I:=1 TO N_VARS DO    BEGIN
       REPEAT
          WRITELN(COUT,'ENTER AGGREGATE VAR # FOR CATA SET VAR',I:5);
          WRITELN(COUT,'OR ENTER 0 IF DATA SET VAR EXCLUDED FROM TEST');
          RST;READLN(QIN,KK);WRITELN(QOUT,KK:5,' CCRRECT?');
       UNTIL ((YES) AND (KK IN (.0..N_VARS.)));
       IF (KK > 0) THEN BEGIN
           N_I_P:=N_I_P+1; I_P(.N_I_P.):=KK;
           V_T(.N_I_P.):=I;  END;  (*IF KK>0 *)
     END;   (* FCR I DO *)
     WRITELN(CP,N_I_P:5);
     FOR I:=1 TC N_I_P DO  WRITE(OP,V_T(.I.):5);
     WRITELN(CF);
END;   (*MED_AG*)
(*-----------------------------------------------------------------
*)
PROCEDURE MECIAN(N_VARS:INT);
BEGIN
     WRITELN(CCUT,'DO YOU WANT TO COMBINE TWO CR MCRE VARIABLES');
     WRITELN(QCUT,'  INTO ONE OR MORE AGGREGATE VARIABLES?');
     IF NCT(YES) TFEN   BEGIN
          WHICH_VEC(N_VARS,N_VARS);
          N_I_P:=0;    END
     ELSE
          MED_AG(N_VARS,I_P,N_I_P);
     NONE_EX;
     R_P(.1.):=L_SIG;
     W_PARAM(N_I_P,1,I_P,R_P);
END;(*MEDIAN*)
(*-------------------------------------------------------------------*)
PROCEDURE KRUWAL(N_VARS:INT);
BEGIN
       WHICH_VEC(N_VARS,N_VARS);
       NONE_EX;
       R_P(.1.):=1;I_P(.1.):=0;
       W_PARAM(0,1,I_P,R_P);
END;  (*KRUWAL*)
(*--------------------------------------------------------------------*)
```

```
PROCEDURE CORREL8(N_VARS,N_EQ_S:INT);
BEGIN
      WHICH_VEC(N_VARS,2);
      EXCLUD_E(N_EQ_S);
      IF  (N_EC_S<40)  THEN I_P(.1.):=1
         ELSE     BEGIN
            WRITELN(QOUT,'THE PROGRAM WILL AUTOMATICALLY COMPUTE');
            WRITELN(QCUT,'SPEARMANS  RHO, AS WELL AS THE PARAMETRIC');
            WRITELN(QCUT,'PEARSONS RHO FROM THE DATA VALUES');
            WRITELN(QCUT,'THE ALGORITHM TO COMPUTE KENDALLS TAU');
            WRITELN(QOUT,'IS SLOW FOR LARGE DATA SETS, SINCE IT IS');
            WRITELN(QOUT,'POLYNOMIAL OF DEGREE 2 (I.E., A DATA SET');
           WRITELN(QOUT,'TWICE AS LONG WILL TAKE 2 SQUARED=4 TIMES');
            WRITELN(QOUT,'AS LONGE TO RUN).  DO YCU WANT TO HAVE');
            WRITELN(QCUT,'KENDALLS TAU COMPUTED?');
              IF YES THEN I_P(.1.):=1 ELSE I_P(.1.):=0;
            END;  (*ELSE*)
      R_P(.1.):=L_SIG;
      W_PARAM(1,1,I_P,R_P);
END;  (*CORREL8*)
(*-----------------------------------------------------------------*)
PROCEDURE QUADE(N_VARS,N_EQ_S:INT);
BEGIN
      WHICH_VEC(N_VARS,N_VARS);
      EXCLUD_E(N_EQ_S);
      N_I_P:=C;N_F_P:=0;I_P(.1.):=0;R_P(.1.):=0;
      W_PARAM(0,0,I_P,R_P);
END;(*QUADE*)
(*----------------------------------------------------------------*)
PROCEDURE REGRET(N_VARS,N_EQ_S:INT);
VAR I,K:INT;  XX:REAL;
BEGIN
  WHICH_VEC(N_VARS,2);
  EXCLUD_B(N_EC_S);
  WRITELN(QOUT,'YCU MAY ENTER THE SLCPE PARAMETER B1:');
  WRITELN(QOUT,'SUCH THAT E(Y)=B1*E(X)+B0');  WRITELN(QOUT,'WHERE');
  WRITELN(QOUT,'Y IS THE DEPENDENT VARIABLE,  X IS INDEPENDENT');
  WRITELN(QOUT,' AND B0 IS THE INTERCEPT');
  WRITELN(QOUT,' CR YOU MAY LET THE THE PROGRAM ESTIMATE B1');
  WRITELN(QOUT,' CO YCU WANT TO ENTER A VALUE FOR B1?');R_P(.1.):=0;
  IF NOT(YES) THEN I_P(.2.):=0 ELSE BEGIN      I_P(.2.):=1;
      REPEAT  RST; WRITELN(QOUT,'ENTER B1');READLN(QIN,XX);
        WRITELN(QCUT,' YOUR B1=',XX:10:3,' CCRRECT?');
      UNTIL YES;  R_P(.1.):=XX;  END; (* IF ENTER OWN B1*)
  WRITELN(QOUT,'IS THE LOWER INDEX VARIABLE Y(DEPENDENT VARIABLE)?');
  IF YES THEN I_P(.1.):=1 ELSE I_P(.1.):=2;R_P(.2.):=0;
  WRITELN(QOUT,'DC YOU WANT TO ESTIMATE Y FOR SOME VALUES OF X?');
  IF NOT(YES) THEN I_P(.3.):=0 ELSE   BEGIN
```

```
      REPEAT RST;WRITELN(QOUT,'HOW MANY VALUES OF X?');
         READLN(QIN,I_P(.3.));WRITELN(QOUT,I_P(.3.):4,' CORRECT?');
      UNTIL YES;
      K:=I_P(.3.);
      FOR I:=1 TO K   DO    BEGIN  RST;
         WRITELN(QOUT,' ENTER X VALUE #',I);
         READLN(QIN,R_P(.(2+I).));   END;  (*FOR I*)
  END; (* ELSE *)
  WRITELN(QOUT,'THE PROGRAM CAN COMPUTE CONFIDENCE INTERVALS FOR');
  WRITELN(QOUT,' THE SLOPE PARAMETER B1; FOR LARGE DATA SETS  IT');
  WRITELN(QOUT,' IS EXPENSIVE--0.5*N**2, N=#PAIRS');R_P(.2.):=0;
  IF (N_EQ_S<61) THEN
      WRITELN(QOUT,' TO DO SO, YOU NEED W VALUE FOR N=',N_EQ_S);
  WRITELN(QOUT,' DO YOU WANT TO FIND CONFIDENCE INTERVALS FOR B1?');
  IF NOT(YES) THEN I_P(.4.):=0 ELSE   BEGIN  I_P(.4.):=1;
    IF (N_EQ_S<61) THEN REPEAT  RST;
      WRITELN(QOUT,' ENTER W FOR N=',N_EQ_S);
      READLN(QIN,R_P(.2.));WRITELN(QOUT,R_P(.2.):10:4,'CORRECT?');
    UNTIL YES;  END;(* IF NOT YES ELSE *)
  K:=I_P(.3.)+2;
  W_PARAM(4,K,I_P,R_P);
END;  (* REGRET *)
(*-------------------------------------------------------------------*)
PROCEDURE SMIRNOV(N_VARS:INT);
BEGIN
      WHICH_VEC(N_VARS,2);
      NONE_EX;R_P(.1.):=0;
   H_A:=' A:NULL HYPOTHESIS OF FORM:  F(X)=G(X)                       ';
   H_B:=' B:NULL H: F(X)=>S(X),  F(X) = 1ST POPULATION DIST';
   H_C:=' C:NULL H: F(X)<=S(X)    AND G(X) =  2D   DIST                ';
   I_P(.1.):=HYP(H_A,H_B,H_C,369);
      W_PARAM(1,0,I_P,R_P);
END;
(*-------------------------------------------------------------------*)
PROCEDURE SQUARANK(N_VARS:INT);
BEGIN
      WHICH_VEC(N_VARS,N_VARS);
      NONE_EX;I_P(.1.):=0;R_P(.1.):=0;
      W_PARAM(0,0,I_P,R_P);
END;
(*-------------------------------------------------------------------*)
PROCEDURE HARTLEY(N_VARS:INT);
BEGIN
      WHICH_VEC(N_VARS,N_VARS);
      NONE_EX;I_P(.1.):=0;R_P(.1.):=0;
      W_PARAM(0,0,I_P,R_P);
END;
(*-------------------------------------------------------------------*)
```

```
PROCEDURE CRAMVONM(N_VARS:INT);
BEGIN
      WHICH_VEC(N_VARS,2);
      NONE_EX;I_P(.1.):=0;R_P(.1.):=0;
      W_PARAM(0,0,I_P,R_P);
END;
(*-----------------------------------------------------------*)
PROCEDURE COCHRAN(N_VARS,N_EQ_S:INT;CELL_CT:BOOLEAN);
VAR IT,N_EX,NB,NT,I,J:INT;ILIST:ARRAY(.1..400.) OF INT;
    HELD:HELDEST;
BEGIN
CH_MIND:=FALSE;PART_VAL(.1.):=0;CELIST(.1.):=0;HELD:=(.255.);
IF NOT CELL_CT THEN BEGIN
  WRITELN(QOUT,'THE COCHRAN TEST USES DICOTOMIZED DATA,I.E.,');
  WRITELN(QOUT,'EACH ELEMENT CAN BE CLASSIFIED AS A SUCCESS(=1).,');
  WRITELN(QOUT,'OR A FAILURE(=0). FOR THE PROGRAM TO COUNT THE');
  WRITELN(QOUT,'SUCCESSES OR FAILURES, EACH TREATMENT OR VARIABLE');
  WRITELN(QOUT,'MUST BE DIVIDIBLE INTO SUCCESSES OR FAILURES BY');
  WRITELN(QOUT,'A SINGLE SCALAR VALUE. E.G., IN TREATMENT ONE');
  WRITELN(QOUT,'ALL VALUES BELOW 19 ARE SUCCESSES, IN TREATMENT');
  WRITELN(QOUT,'TWO ALL VALUES ABOVE 27 ARE SUCCESSES,ETC. IF');
  WRITELN(QOUT,'YOUR DATA SET IS NOT DIVISIBLE IN THIS MANNER, OR');
  WRITELN(QOUT,'YOU DON"T WANT TO ENTER THIS WAY, YOU CAN ENTER ');
  WRITELN(QOUT,'WHETHER A BLOCK/TREATMENT IS A SUCCESS OR FAILURE');
  WRITELN(QOUT,'ONE CELL AT A TIME.  DO YOU WANT TO CHANGE TO THIS');
  WRITELN(QOUT,'LATTER METHOD, THAT IS, ENTERING SUCCESS OR FAILURE');
  WRITELN(QOUT,'ONE CELL AT A TIME?');CH_MIND:=YES;
  IF NOT CH_MIND THEN BEGIN
     N_V_T:=0; IT:=0;
     FOR J:=1 TO N_VARS DO BEGIN
       WRITELN(QOUT,'IS TREATMENT',J:3,' IN TEST?');
       IF YES THEN BEGIN
         N_V_T:=N_V_T+1;V_T(.N_V_T.):=J; IT:=IT+1;
         REPEAT
           WRITELN(QOUT,'ENTER NUMBER WHICH DIVIDES TREATMENT',J:5);
           WRITELN(QOUT,'INTO SUCCESSES OR FAILURES:');
           RST;READLN(QIN,XX);WRITELN(QOUT,XX:10:3,'  CORRECT?');
         UNTIL YES;    PART_VAL(.(IT*2)-1.):=XX;
         REPEAT
           WRITELN(QOUT,'ARE VALUES BELOW',XX:10:3,' SUCCESSES?');
           IF YES THEN YY:=1.0 ELSE YY:=-1.0;PART_VAL(.IT*2.):=YY;
           IF(YY<0)THEN WRITELN(QOUT,'FOR TREATMENT',J:4,
           '  VALUES ABOVE',PART_VAL(.(IT*2)-1.):10:3,' ARE SUCCESSES');
           IF(YY>0)THEN WRITELN(QOUT,'FOR TREATMENT',J:4,
           '  VALUES BELOW',PART_VAL(.(IT*2)-1.):10:3,' ARE SUCCESSES');
           WRITELN(QOUT,'CORRECT?');
         UNTIL YES;
       END; (* IF I IS IN TEST *)
```

```
     END;     (*FOR I=1,N_VARS *)
     W_CELLS(73,N_EQ_S,IT,(2*IT),PART_VAL, CELIST);
     WRITELN(CP,N_V_T:5);
     FOR I:=1 TC N_V_T DO    BEGIN
          WRITE(OF,V_T(.I.):5);
          IF ((I=IT) OR (0=(I MOD 16))) THEN WRITELN(OP); END;
     EXCLUD E(N_EC_S);
  END;(*IF NCT CHANGE MIND*)
END; (*IF NOT CELL_CT*)
IF (CELL_CT CR CH_MINC) THEN     BEGIN    REPEAT
      WRITELN(COUT,'ENTER NUMBER OF BLOCKS AND OF TREATMENTS');
      RST;READLN(CIN,NB,NT);WRITELN(QOUT,'NUMBER CF BLOCKS:',NB:5);
      WRITELN(COUT,'NUMBER OF TREATMENTS:',NT:5,'    CORRECT?');
      UNTIL YES;
      FOR I:=1 TO (NB+NT) DO CELIST(.I.):=0;
      FOR I:=1 TO NB DO
         FOR J:=1 TO NT DO BEGIN    K:=(I-1)*NT+J;
            WRITELN(QCUT,'IS BLOCK',I:3,' TREATMENT',J:3,' SUCCESS?');
           IF YES THEN ILIST(.K.):=1 ELSE ILIST(.K.):=0; END;(*FOR J*)
      REPEAT    (*UNTIL ACCEPT THE 1/0 MATRIX*)
         WRITELN(COUT,'BLOCK*TREAMENT MATRIX,1=SUCCESS:');
         FOR I:=1 TO NB DO BEGIN  WRITE(QOUT,I:4,'  ');
              FOR J:=1 TC (NT-1) DO
                  WRITE(QOUT,ILIST(.(J+(I-1)*NT).):3);
              WRITELN(QOUT,ILIST(.(I*NT).):3); END;(*FOR I*)
         WRITELN(COUT,' CORRECT?');OKAY:=YES;
         IF NCT OKAY THEN BEGIN
              WRITELN(QOUT,'ENTER INDEX OF BLOCK TO BE CORRECTED:');
              RST;READLN(QIN,I);WRITELN(CCUT,'CURRENT BLOCK',I:5);
              WRITE(QOUT,I:4,'  ');
              FOR J:=1 TC (NT-1) DO
                  WRITE(QOUT,ILIST(.(J+(I-1)*NT).):3);
              WRITELN(QOUT,ILIST(.(I*NT).):3);
              FOR J:=1 TC NT DO BEGIN    K:=(I-1)*NT+J;
                    WRITELN(QOUT,' TREATMENT',J:3,' SUCCESS?');
                    IF YES THEN ILIST(.K.):=1 ELSE ILIST(.K.):=0;
                   WRITELN(QCUT,ILIST(.K.):5);
         END;(*FOR J*)ENC;(* IF NOT OKAY *)
      UNTIL  CKAY;
      FOR I:=1 TO NB DO
         FOR J:=1 TO NT CO BEGIN K:=(I-1)*NT+J;
              CELIST(.I.):=CELIST(.I.)+ILIST(.K.);
              CELIST(.(J+NB).):=CELIST(.(J+NB).)+ILIST(.K.);
          ENC;(*FCR J DO*)
          IF NOT CH_MIND THEN TYP_CEL:=71;
          IF CH_MIND THEN TYP_CEL:=72;
       W_CELLS(TYF_CEL,NB,NT,(NB+NT),PART_VAL, CELIST);
END;(*IF CELL_CT*)
```

```
R_P(.1.):=L_SIG;I_P(.1.):=0;
W_PARAM(0,1,I_P,R_P);
END; (*COCHRAN*)
(*-----------------------------------------------------------*)
PROCEDURE WILCOXON(N_VARS,N_EQ_S:INT);
VAR  SHIFT:REAL;
BEGIN
   WRITELN(QOUT,'THE WILCOXON SIGNED RANK TEST WILL EVALUATE THE');
   WRITELN(QOUT,'DIFFERENCES BETWEEN TWO EQUAL LENGTH VARIABLES');
   WRITELN(QOUT,'THIS PROGRAM SUBTRACTS THE VALUES OF THE VARIABLE');
   WRITELN(QOUT,'WITH THE SMALLER INDEX FROM THE VALUES OF THE ');
   WRITELN(QOUT,'VARIABLE WITH THE LARGER INDEX.  THE PROGRAM');
   WRITELN(QOUT,'ALSO ALLOWS ADDING A CONSTANT TO THE HIGHER INDEX');
   WRITELN(QOUT,'VARIABLE, WHICH ALLOWS FLEXIBILITY IN DATA');
   WRITELN(QOUT,'ANALYSIS.  E.G., IF YOU CAN REJECT THE NULL HYP');
   WRITELN(QOUT,'E(X)>=E(Y), IMPLYING THAT E(X)<E(Y), WILL YOU ');
   WRITELN(QOUT,'STILL REJECT IF YOU ADD 200 TO EACH VALUE OF X?');
   WRITELN(QOUT,'THE DEFAULT VALUE OF THE CONSTANT=0');
   WRITELN(QOUT);
   WRITELN(QOUT,'DO YOU WANT TO ADD A CONSTANT TO THE HIGHER INDEX');
   WRITELN(QOUT,'    VARIABLE');
   IF YES THEN     REPEAT    RST;
         WRITELN(QOUT,'ENTER CONSTANT');READLN(QIN,R_P(.1.));
         WRITELN(QOUT,R_P(.1.):10:3,' CORRECT?')   UNTIL YES
      ELSE  R_P(.1.):=0.0;
  WHICH_VEC(N_VARS,2);
  EXCLUD_B(N_EQ_S);
  WRITELN(QOUT,' X IS THE LOWER INDEX VARIABLE');
   H_A:=' A:NULL HYP HO: E(X)=E(Y),ALT HYP H1: E(X)<>E(Y)  ';
   H_B:=' B:NULL HYP HO: E(X)>=E(Y),ALT HYP H1: E(X)<E(Y)  ';
   H_C:=' C:NULL HYP HO: E(X)<=E(Y),ALT HYP H1: E(X)>E(Y)  ';
   I_P(.1.):=HYP(H_A,H_B,H_C,281);
  R_P(.2.):=L_SIG;
  W_PARAM(1,2,I_P,R_P);
END; (*WILCOXON*)
(*-----------------------------------------------------------*)
PROCEDURE MCNEMAR(N_EQ_S,N_VARS:INT;EQUAL_S,CELL_CTR:BOOLEAN);
VAR TWO,BEFORE,AFTER,I,J:INT;
      S_B,S_A:REAL;
      A_SU_B,B_SU_B,CH_MIND:BOOLEAN;
BEGIN    CH_MIND:=FALSE;TWO:=2;
      PART_VAL(.1.):=0;CELIST(.1.):=0;
WRITELN(QOUT,' THIS TEST REQUIRES COUNTS IN FOUR CELLS:');
WRITELN(QOUT,'TWO DIMENSIONS WITH TWO LEVELS FOR EACH DIMENSION');
WRITELN(QOUT,'FOR EXAMPLE, IF BEFORE/AFTER ARE THE DIMENSIONS, AND');
WRITELN(QOUT,'SUCCESS/FAILURE THE LEVELS, # MARKS THE CELLS:');
WRITELN(QOUT,'                           AFTER:         ');
WRITELN(QOUT,'                 SUCCESS           FAILURE   ');
```

```
WRITELN(QOUT,'BEFORE:   ------->-------------------------------------');
WRITELN(QOUT,'  SUCCESS  | #SUCCESS-SUCCESS | #SUCCESS-FAILURE  |');
WRITELN(QOUT,'           |------------------|------------------|');
WRITELN(QOUT,'  FAILURE  | #FAILURE-SUCCESS | #FAILURE-FAILURE  |');
WRITELN(QOUT,'           --------------------------------------');
IF NOT CELL_CTR THEN BEGIN
  IF (N_VARS=1) OR NOT EQUAL_S THEN   BEGIN
      WRITELN(QOUT,' THE DATA SET YOU SPECIFIED CANNOT HAVE THE');
      WRITELN(QOUT,'TEST PERFORMED BECAUSE:');
        IF (N_VARS=1) THEN WRITELN(QOUT,
            'TWO VARIABLES ARE NEEDED BUT THE DATA SET HAS ONLY ONE ');
        IF (N_VARS=1) AND NOT EQUAL_S THEN WRITELN(QOUT,'  AND,');
        IF NOT EQUAL_S THEN WRITELN(QOUT,
              'THIS TEST REQUIRES PAIRS--VARIABLES OF EQUAL LENGTH');
          WRITELN(QOUT,' YOU WILL HAVE A CHANCE TO DIRECTLY ENTER',
              ' CELL COUNTS'); CH_MIND:=TRUE;
     END;(*IF N_VARS=1 OR NOT EQUAL_S*)
   IF (N_VARS>=2) AND EQUAL_S THEN BEGIN
      WRITELN(QOUT,' TO HAVE THE PROGRAM AUTOMATICALLY DERIVE');
WRITELN(QOUT,' CELL COUNTS, YOU WILL HAVE TO SELECT TWO VARIABLES');
WRITELN(QOUT,'ONE AS "BEFORE",THE OTHER "AFTER".  THEN YOU WILL HAVE');
WRITELN(QOUT,'TO ENTER A SINGLE SCALAR VALUE THAT DIVIDES THE FIRST');
WRITELN(QOUT,'VARIABLE INTO SUCCESSES AND FAILURES,AND A SECOND');
WRITELN(QOUT,'SCALAR  VALUE TO SO DIVIDE THE SECOND VARIABLE.');
WRITELN(QOUT,'IF THIS IS INCONVENIENT OR IMPOSSIBLE, YOU MAY SIMPLY');
WRITELN(QOUT,'ENTER THE CELL COUNTS YOURSELF.');WRITELN(QOUT);
WRITELN(QOUT,'DO YOU WANT TO ENTER THE CELL COUNTS YOURSELF?');
 CH_MIND:=YES;
 IF NOT CH_MIND THEN BEGIN   REPEAT
         WRITELN(QOUT,' OF',N_VARS:3, ' VARIABLES, WHICH IS BEFORE');
         RST;READLN(QIN,BEFORE);
         WRITELN(QOUT,' WHICH IS AFTER?');RST;READLN(QIN,AFTER);
         WRITELN(QOUT,' WHAT IS SCALAR VALUE TO DIVIDE BEFORE?');
             RST;READLN(QIN,S_B);
         WRITELN(QOUT,' ARE SUCCESSES LESS THAN THAT VALUE?');
             B_SU_B:=YES;
        WRITELN(QOUT,' WHAT IS SCALAR VALUE TO DIVIDE AFTER?');
         RST;READLN(QIN,S_A);
        WRITELN(QOUT,' ARE SUCCESSES LESS THAN THAT VALUE?');
         A_SU_B:=YES;
        IF B_SU_B THEN WRITELN(QOUT,' BEFORE: FOR VAR',BEFORE:3,
            ' SUCCESSES ARE BELOW ',S_B:10:3);
        IF NOT B_SU_B THEN WRITELN(QOUT,' BEFORE: FOR VAR',BEFORE:3,
            ' SUCCESSES ARE ABOVE ',S_B:10:3);
        IF A_SU_B THEN WRITELN(QOUT,' AFTER: FOR VAR',AFTER:3,
            ' SUCCESSES ARE BELOW ',S_A:10:3);
        IF NOT A_SU_B THEN WRITELN(QOUT,' AFTER: FOR VAR',AFTER:3,
            ' SUCCESSES ARE ABOVE ',S_A:10:3);
```

```
       WRITELN(QCLT,' CORRECT?')
  UNTIL  YES; (*REPEAT UNTIL*)
  I:=0; IF (BEFORE>AFTER) THEN I:=1;
      PART_VAL(.1+I.):=S_B; PART_VAL(.2-I.):=S_A;
      IF B_SU_B THEN XX:=1.0 ELSE XX:=-1.0; PART_VAL(.3+I.):=XX;
      IF A_SU_B THEN XX:=1.0 ELSE XX:=-1.0; PART_VAL(.4-I.):=XX;
      W_CELLS(63,2,2,5,PART_VAL,CELIST);
     PART_VAL(.5.):=1.0-I; WRITELN(OP,TWO:5);(*IN LIEU OF WHICH_VEC*)
      IF (I=0) THEN  WRITELN(OP,BEFORE:5,AFTER:5)
         ELSE WRITELN(OP,AFTER:5,BEFORE:5);
      EXCLUD_B(N_EQ_S);
   END;(* IF NCT CH_MIND *)
 END; (* IF N_VARS>1 AND EQUAL_S *)
END; (*IF NOT CELL_CTR *)
IF CELL_CTR CR CH_MINC THEN BEGIN
   REPEAT
      WRITELN(COUT,' ENTER THE COUNT OF SUCCESS-SLCCESS CELL:');
         RST; REACLN(CIN,CELIST(.1.));
      WRITELN(QOUT,' ENTER COUNT OF SUCCESS-FAILURE CELL:');
         RST; REACLN(CIN,CELIST(.2.));
      WRITELN(COUT,' ENTER CCUNT OF FAILURE-SUCCESS CELL:');
         RST; REACLN(CIN,CELIST(.3.));
      WRITELN(COUT,' ENTER CCUNT OF FAILURE-FAILURES CELL:');
         RST; REACLN(QIN,CELIST(.4.));
      WRITELN(COUT,'                         AFTER:');
      WRITELN(COUT,'BEFORE:         SUCCESS    FAILURE');
      WRITELN(QOUT,'SUCCESS  ',CELIST(.1.):10,CELIST(.2.):10);
      WRITELN(COUT,'FAILURE  ',CELIST(.3.):10,CELIST(.4.):10);
      WRITELN(QOUT);WRITELN(QOUT,'CORRECT? ');
   UNTIL YES;
  IF NOT CH_MIND THEN TYP_CEL:=61;
  IF CH_MIND THEN TYP_CEL:=62;
  W_CELLS(TYP_CEL,2,2,4,PART_VAL,CELIST);
END;(* IF CELL_CTR OR CH_MIND *)
N_R_P:=1;N_I_P:=0;R_P(.1.):=L_SIG;I_P(.1.):=0;
W_PARAM(0,I,I_F,R_P);
END;(* MCNEMAR *)
(*--------------------------------------------------------*)
PROCEDURE MCNEMAR1;
BEGIN
      MCNEMAR(0,2,FALSE,TRUE);
END;
(*------------------------------------------------------*)
PROCEDURE E_SIZE(N_VARS:INTEGER;VAR_SIZE:VECTORI;
     VAR EQUAL_S:BOOLEAN;VAR N_EQ_S:INT);
   (*CHECKS TC SEE IF ALL VARIABLES OR TREATMENT HAVE EQUAL NUMBER
      OF ELEMENTS: IF SO, REPORTS THE NUMBER OF ELEMENTS*)
VAR I,J,K:INTEGER;
```

```
BEGIN
 EQUAL_S:=TRUE;N_EQ_S:=0;
 IF (N_VARS>1) THEN BEGIN
   FOR I:=2 TO N_VARS DO
     IF (VAR_SIZE(.(I-1).)<>VAR_SIZE(.I.)) THEN EQUAL_S:=FALSE;
 IF EQUAL_S THEN WRITELN(QOUT,'THE VARIABLES/TREATMENTS ',
     'ARE OF EQUAL SIZE: PAIRED COMPARISON TESTS ARE POSSIBLE');
 IF NOT EQUAL_S THEN WRITELN(QOUT,'THE VARIABLES/TREATEMENTS ',
 'ARE NOT OF EQUAL SIZE: PAIRED COMPARISON TESTS ARE NOT POSSIBLE');
 IF EQUAL_S THEN N_EQ_S:=VAR_SIZE(.1.);
  END;
END;
(*------------------------------------------------------------------*)
PROCEDURE WILKES(IN_V_T:INT);
VAR Z:R_PAR; I:INT;
BEGIN
    K:=IN_V_T DIV 2; I_P(.1.):=K;N_I_P:=1;
    CASE IN_V_T OF
 30: BEGIN I:=15;Z(.1.):=0.4255;Z(.2.):=0.2944;Z(.3.):=0.2487;
 Z(.4.):=0.2148;Z(.5.):=0.1870;Z(.6.):=0.1630;Z(.7.):=0.1415;
 Z(.8.):=0.1219;Z(.9.):=0.1036;Z(.10.):=0.0862;Z(.11.):=0.0697;
 Z(.12.):=0.0537;Z(.13.):=0.0381;Z(.14.):=0.0227;Z(.15.):=0.0076;END;
 29: BEGIN I:=15;Z(.1.):=0.4291;Z(.2.):=0.2968;Z(.3.):=0.2499;
 Z(.4.):=0.2150;Z(.5.):=0.1864;Z(.6.):=0.1616;Z(.7.):=0.1395;
 Z(.8.):=0.1192;Z(.9.):=0.1002;Z(.10.):=0.0822;Z(.11.):=0.0650;
 Z(.12.):=0.0483;Z(.13.):=0.0320;Z(.14.):=0.0084;Z(.15.):=0.0;END;
 28: BEGIN I:=14;Z(.1.):=0.4328;Z(.2.):=0.2992;Z(.3.):=0.2510;
 Z(.4.):=0.2151;Z(.5.):=0.1857;Z(.6.):=0.1601;Z(.7.):=0.1372;
 Z(.8.):=0.1162;Z(.9.):=0.0965;Z(.10.):=0.0778;Z(.11.):=0.0598;
 Z(.12.):=0.0424;Z(.13.):=0.0253;Z(.14.):=0.0084;END;
 27: BEGIN I:=14;Z(.1.):=0.4366;Z(.2.):=0.3018;Z(.3.):=0.2522;
 Z(.4.):=0.2152;Z(.5.):=0.1848;Z(.6.):=0.1584;Z(.7.):=0.1346;
 Z(.8.):=0.1162;Z(.9.):=0.0923;Z(.10.):=0.0728;Z(.11.):=0.0540;
 Z(.12.):=0.0358;Z(.13.):=0.0178;Z(.14.):=0.0;END;
 26: BEGIN I:=13;Z(.1.):=0.4407;Z(.2.):=0.3043;Z(.3.):=0.2533;
 Z(.4.):=0.2151;Z(.5.):=0.1836;Z(.6.):=0.1563;Z(.7.):=0.1316;
 Z(.8.):=0.1089;Z(.9.):=0.0876;Z(.10.):=0.0672;Z(.11.):=0.0476;
 Z(.12.):=0.0284;Z(.13.):=0.0094;END;
 25: BEGIN I:=13;Z(.1.):=0.4450;Z(.2.):=0.3069;Z(.3.):=0.2543;
 Z(.4.):=0.2148;Z(.5.):=0.1822;Z(.6.):=0.1539;Z(.7.):=0.1283;
 Z(.8.):=0.1046;Z(.9.):=0.0823;Z(.10.):=0.0610;Z(.11.):=0.0403;
 Z(.12.):=0.0200;Z(.13.):=0.0;END;
 24: BEGIN I:=12;Z(.1.):=0.4493;Z(.2.):=0.3098;Z(.3.):=0.2554;
 Z(.4.):=0.2145;Z(.5.):=0.1807;Z(.6.):=0.1512;Z(.7.):=0.1245;
 Z(.8.):=0.0997;Z(.9.):=0.0764;Z(.10.):=0.0539;Z(.11.):=0.0321;
 Z(.12.):=0.0107;END;
 23: BEGIN I:=12;Z(.1.):=0.4542;Z(.2.):=0.3126;Z(.3.):=0.2563;
 Z(.4.):=0.2139;Z(.5.):=0.1787;Z(.6.):=0.1480;Z(.7.):=0.1201;
```

```
Z(.8.):=0.0941;Z(.9.):=0.0696;Z(.10.):=0.0459;Z(.11.):=0.0228;
Z(.12.):=0.0;END;
22:  BEGIN I:=11;Z(.1.):=0.4590;Z(.2.):=0.3156;Z(.3.):=0.2573;
Z(.4.):=0.2131;Z(.5.):=0.1764;Z(.6.):=0.1443;Z(.7.):=0.1150;
Z(.8.):=0.0878;Z(.9.):=0.0618;Z(.10.):=0.0368;Z(.11.):=0.0122;END;
21:  BEGIN I:=11;Z(.1.):=0.4643;Z(.2.):=0.3185;Z(.3.):=0.2578;
Z(.4.):=0.2119;Z(.5.):=0.1736;Z(.6.):=0.1399;Z(.7.):=0.1092;
Z(.8.):=0.0804;Z(.9.):=0.0530;Z(.10.):=0.0263;Z(.11.):=0.0000;END;
20:  BEGIN I:=10;Z(.1.):=0.4734;Z(.2.):=0.3211;Z(.3.):=0.2565;
Z(.4.):=0.2085;Z(.5.):=0.1686;Z(.6.):=0.1334;Z(.7.):=0.1013;
Z(.8.):=0.0711;Z(.9.):=0.0422;Z(.10.):=0.0140;END;
19:  BEGIN I:=10;Z(.1.):=0.4808;Z(.2.):=0.3232;Z(.3.):=0.2561;
Z(.4.):=0.2059;Z(.5.):=0.1641;Z(.6.):=0.1271;Z(.7.):=0.0932;
Z(.8.):=0.0612;Z(.9.):=0.0303;Z(.10.):=0.0000;END;
18:  BEGIN I:=9;Z(.1.):=0.4886;Z(.2.):=0.3253;Z(.3.):=0.2553;
Z(.4.):=0.1587;Z(.5.):=0.1587;Z(.6.):=0.1197;Z(.7.):=0.0837;
Z(.8.):=0.0496;Z(.9.):=0.0163;END;
17:  BEGIN I:=9;Z(.1.):=0.4968;Z(.2.):=0.3273;Z(.3.):=0.2540;
Z(.4.):=0.1988;Z(.5.):=0.1524;Z(.6.):=0.1109;Z(.7.):=0.0725;
Z(.8.):=0.0359;Z(.9.):=0.0000;END;
16:  BEGIN I:=8;Z(.1.):=0.5056;Z(.2.):=0.3290;Z(.3.):=0.2521;
Z(.4.):=0.1939;Z(.5.):=0.1447;Z(.6.):=0.1005;Z(.7.):=0.0593;
Z(.8.):=0.0196;END;
15:  BEGIN I:=8;Z(.1.):=0.5150;Z(.2.):=0.3306;Z(.3.):=0.2495;
Z(.4.):=0.1878;Z(.5.):=0.1353;Z(.6.):=0.0880;Z(.7.):=0.0433;
Z(.8.):=0.0;END;
14:  BEGIN I:=7;Z(.1.):=0.5251;Z(.2.):=0.3318;Z(.3.):=0.2460;
Z(.4.):=0.1802;Z(.5.):=0.1240;Z(.6.):=0.0727;Z(.7.):=0.0240;END;
13:  BEGIN I:=7;Z(.1.):=0.5359;Z(.2.):=0.3325;Z(.3.):=0.2412;
Z(.4.):=0.1707;Z(.5.):=0.1099;Z(.6.):=0.0539;Z(.7.):=0.0;END;
12:  BEGIN I:=6;Z(.1.):=0.5475;Z(.2.):=0.3325;Z(.3.):=0.2347;
Z(.4.):=0.1586;Z(.5.):=0.0922;Z(.6.):=0.0303;END;
11:  BEGIN I:=6;Z(.1.):=0.5601;Z(.2.):=0.3315;Z(.3.):=0.2260;
Z(.4.):=0.1429;Z(.5.):=0.0695;Z(.6.):=0.0;END;
10:  BEGIN I:=5;Z(.1.):=0.5739;Z(.2.):=0.3291;Z(.3.):=0.2141;
Z(.4.):=0.1224;Z(.5.):=0.0399;END;
 9:  BEGIN I:=5;Z(.1.):=0.5888;Z(.2.):=0.3244;Z(.3.):=0.1976;
Z(.4.):=0.0947;Z(.5.):=0.0;END;
 8:  BEGIN I:=4;Z(.1.):=0.6052;Z(.2.):=0.3164;Z(.3.):=0.1743;
Z(.4.):=0.0561;END;
 7:  BEGIN I:=4;Z(.1.):=0.6233;Z(.2.):=0.3031;Z(.3.):=0.1401;
Z(.4.):=0.0;END;
 6:  BEGIN I:=3;Z(.1.):=0.6431;Z(.2.):=0.2806;Z(.3.):=0.0875;END;
 5:  BEGIN I:=3;Z(.1.):=0.6646;Z(.2.):=0.2413;Z(.3.):=0.0;END;
 4:  BEGIN I:=2;Z(.1.):=0.6872;Z(.2.):=0.1667;END;
 3:  BEGIN I:=2;Z(.1.):=0.7071;Z(.2.):=0.0;END;
 2:  BEGIN I:=1;Z(.1.):=0.7071;END;
END;  (*CASE*)
```

```
N_R_P:=I;
W_PARAM(N_I_P,N_R_P,I_P,Z);
END; (*WILKES*)
(*------------------------------------------------------------------*)
PROCEDURE LILLIE;
VAR I:INT;
BEGIN
    WRITELN(QOUT,'CHOOSE BETWEEN THE FOLLOWING DISTRUBUTIONS:');
    REPEAT  RST;
       WRITELN(QOUT,' 1. NORMAL');
       WRITELN(QOUT,' 2. EXPONENTIAL');
       READLN(QIN,I);
    UNTIL (I IN (.1,2.));
   H_A:=' A:NULL HYPOTHESIS OF FORM: F*(X)=S(X)                ';
   H_B:=' B:NULL H: F*(X)=>S(X), F*(X) IS SPEC THEOR DIST      ';
   H_C:=' C:NULL H: F*(X)<=S(X)   AND S(X) IS EMP CDF          ';
   I_P(.2.):=HYP(H_A,H_B,H_C,347);
    I_P(.1.):=I;R_P(.1.):=0;
    W_PARAM(2,C,I_P,R_P);
END; (*LILLIE *)
(*----------------------------------------------------------------*)
PROCEDURE UPAR(VAR R_P:R_PAR);
VAR X,Y:REAL;
BEGIN
    REPEAT
        WRITELN(QOUT,'ENTER LOWER LIMIT A AND UPPER LIMIT B');
        RST; READ(QIN,X,Y);
        WRITELN(QOUT,'A=',X:10:3,' B=',Y:10:3,' CORRECT?');
    UNTIL YES;
    R_P(.1.):=X;R_P(.2.):=Y;N_R_P:=2;
END;  (*UPAR*)
(*----------------------------------------------------------------*)
PROCEDURE WPAR(VAR R_P:R_PAR);
VAR X,Y:REAL;
BEGIN
    REPEAT
        WRITELN(QOUT,'WITH CDF OF FORM:');
        WRITELN(QOUT,'F(X)=1 - EXP(-(LAMBDA*X)**ALPHA)');
        WRITELN(QOUT,'ENTER SHAPE PARAMETER ALPHA, AND');
        WRITELN(QOUT,'   RATE PARAMETER LAMBDA:');
        RST; READ(QIN,X,Y);
        WRITELN(QOUT,'ALPHA=',X:10:3,' LAMBDA=',Y:10:3,' CORRECT?');
    UNTIL YES;
    R_P(.1.):=X;R_P(.2.):=Y;N_R_P:=2;
END;  (*WPAR*)
(*----------------------------------------------------------------*)
PROCEDURE ERPAR(VAR R_P:R_PAR;VAR I_P:I_PAR);
VAR Y:REAL; I:INT;
```

```
BEGIN
    REPEAT
        WRITELN(QOUT,'ENTER INTEGER SHAPE PARAMETER N, AND');
        WRITELN(QOUT,'    RATE PARAMETER LAMBDA:');
        RST; READ(QIN,I,Y);
        WRITELN(QOUT,'N=',I:5,' LAMBDA=',Y:10:3,' CORRECT?');
    UNTIL YES;
    R_P(.1.):=Y;I_P(.3.):=I;N_R_P:=1;N_I_P:=2;
END;  (*ERPAR*)
(*-------------------------------------------------------------------*)
PROCEDURE NPAR(VAR R_P:R_PAR);
VAR X,Y:REAL;
BEGIN
    REPEAT
        WRITELN(QOUT,'ENTER PARAMETER MU--E(X)--AND');
        WRITELN(QOUT,'    SIGMA--VAR(X):');
        RST; READ(QIN,X,Y);
        WRITELN(QOUT,'MU=',X:10:3,' SIGMA=',Y:10:3,' CORRECT?');
    UNTIL YES;
    R_P(.1.):=X;R_P(.2.):=Y;N_R_P:=2;
END;  (*NPAR*)
(*-------------------------------------------------------------------*)
PROCEDURE EXPAR(VAR R_P:R_PAR);
VAR X,Y:REAL;
BEGIN
    REPEAT
        WRITELN(QOUT,'ENTER RECIPROCAL OF E(X), I.E.,');
        WRITELN(QOUT,'    RATE PARAMETER LAMBDA:');
        RST; READ(QIN,X);
        WRITELN(QOUT,'LAMBDA=',X:10:3,' CORRECT?');
    UNTIL YES;
    R_P(.1.):=X;N_R_P:=1;
END;  (*EXPAR*)
(*-------------------------------------------------------------------*)
PROCEDURE KOLMOG;
VAR I:INT;
BEGIN
    WRITELN(QOUT,'CHOOSE AMONG THE FOLLOWING DISTRIBUTIONS:');
    REPEAT
        WRITELN(QOUT,'1. NORMAL');
        WRITELN(QOUT,'2. EXPONENTIAL');
        WRITELN(QOUT,'3. UNIFORM <CONTINUOUS>');
        WRITELN(QOUT,'4. ERLANG <GAMMA WITH INTEGER SHAPE PARAM>');
        WRITELN(QOUT,'5. WEIBULL');
        RST; WRITELN(QOUT,' ENTER CHOICE, 1-5');
        READLN(QIN,I); WRITELN(QOUT,I:5,' CORRECT?');
    UNTIL ((YES) AND (I IN (.1..5.)));
    CASE I OF
```

```
          1: NPAR(R_P);
          2: EXPAR(R_P);
          3: UPAR(R_P);
          4: ERPAR(R_P,I_P);
          5: WPAR(R_P);
    END;  (*CASE*)
    I_P(.1.):=I;
   H_A:=' A:NULL HYPOTHESIS OF FORM: F*(X)=S(X)                  ';
   H_B:=' B:NULL H: F*(X)=>S(X), F*(X) IS SPEC THEOR DIST  ';
   H_C:=' C:NULL H: F*(X)<=S(X)    AND S(X) IS EMP CDF       ';
   I_P(.2.):=HYP(H_A,H_B,H_C,347);
    IF (I=4) THEN N_I_P:=3 ELSE N_I_P:=2;
    IF ((I=2) OR (I=4)) THEN N_R_P:=1 ELSE N_R_P:=2;
    W_PARAM(N_I_P,N_R_P,I_P,R_P);
END; (*KOLMOG*)
(*-------------------------------------------------------------*)
PROCEDURE GDFIT_W(VAR R_P:R_PAR;VAR I_P:I_PAR);
VAR X,Y:REAL;
BEGIN
        N_R_P:=0;R_P(.1.):=0;
        WRITELN(QOUT,'WITH CDF OF FORM:');
        WRITELN(QOUT,'F(X)=1 - EXP(-(LAMBDA*X)**ALPHA)');
        I_P(.2.):=0;
        REPEAT
            WRITELN(QOUT,'ENTER SHAPE PARAMETER ALPHA, AND');
            WRITELN(QOUT,'    RATE PARAMETER LAMBDA:');
            RST; READ(QIN,X,Y); WRITELN(QOUT,
          'ALPHA=',X:10:3,' LAMBDA=',Y:10:3,' CORRECT?');
        UNTIL YES;
        R_P(.1.):=X;R_P(.2.):=Y;N_R_P:=2;
END;  (*GDFIT_W*)
(*-------------------------------------------------------------*)
PROCEDURE GDFIT_ER(VAR I_P:I_PAR;VAR R_P:R_PAR);
VAR BETA,N,I,J:INT;SPEC:BOOLEAN;ALPHA:REAL;X:REAL;
BEGIN
   WRITELN(QOUT,'N IS THE INTEGER SHAPE PARAMETER, AND LAMBDA');
  WRITELN(QOUT,'IS THE RATE OR SCALE PARAMETER.  YOU MUST SPECIFY');
  WRITELN(QOUT,'N; YOU CAN CHOOSE TO ENTER A LAMBDA PARAMETER');
  WRITELN(QOUT,'OR ALLOW THE PROGRAM TO ESTIMATE IT.');
   WRITELN(QOUT,'DO YOU WISH TO ENTER A LAMBDA PARAMETER?');
    IF YES THEN
        REPEAT  RST; I_P(.2.):=0;
           WRITELN(QOUT,' ENTER LAMBDA');READLN(QIN,X);
           WRITELN(QOUT,'LAMBDA=',X:10:3,' CORRECT?');
           R_P(.1.):=X;       UNTIL YES
    ELSE  I_P(.2.):=1;
    REPEAT
       RST; WRITELN(QOUT,'ENTER SHAPE PARAMETER N, INTEGER > 1');
```

```
     READLN(QIN,N);WRITELN(QOUT,'N= ',N:5,' CORRECT?');
  UNTIL ((YES) AND (N >1));
  I_P(.3.):=N;
END; (*GDFIT_ER*)
(*-----------------------------------------------------------------*)
PROCEDURE GD_FIT(N_VARS: INT);
VAR    I,J:INTEGER;
BEGIN
   FOR I:=1 TO 3 DO BEGIN I_P(.I.):=0; R_P(.I.):=0.0; END;(*FOR I*)
   WRITELN(QOUT,'CHOOSE AMONG FIVE THEORETICAL DISTRIBUTIONS: ');
   WRITELN(QOUT,'  1. NORMAL                                  ');
   WRITELN(QOUT,'  2. UNIFORM                                ');
   WRITELN(QOUT,'  3. EXPONENTIAL                             ');
   WRITELN(QOUT,'  4. ERLANG (GAMMA WITH INTEGER BETA PARAMETER');
   WRITELN(QOUT,'  5. WEIBULL');
     REPEAT   RST;
       WRITELN(QOUT,' WHICH DISTRIBUTION (1-5)');
       READLN(QIN,I_P(.1.)); UNTIL (I_P(.1.) IN (.1..5.));
     CASE (I_P(.1.)) OF
       1: GDFIT_N(I_P,R_P);
       2: GDFIT_U(R_P);
       3: GDFIT_EX(I_P,R_P);
       4: GDFIT_ER(I_P,R_P);
       5: GDFIT_W(R_P,I_P);
     END;(*CASE*)
     R_P(.3.):=L_SIG;
     W_PARAM(3,3,I_P,R_P);
END; (*GD_FIT*)
(*-----------------------------------------------------------------*)
PROCEDURE CHECK_N(N_VARS:INT;VAR N_TEST,IN_V_T:INT;N_ELMS:VECTORI);
(* THIS PROCEDURE SCREENS TO MAKE SURE THAT THE GOODNESS OF FIT
 TEST GETS AT LEAST 15 DATA ELEMENTS, AND THAT KOLMOGOROV, LILLIEFORS,
 AND SHAPIRO-WILKES TEST GET NO MORE THAN 30 *)
VAR ONE:INT; G_V:BOOLEAN;
BEGIN
  ONE:=1;
  IF (N_VARS=1) THEN N_V_T:=1 ELSE
   BEGIN
     WRITELN(QOUT,'OF THE',N_VARS:3,' VARS IN THE DATA SET');
     WRITELN(QOUT,'WHICH ONE IS USED IN THIS TEST?');
     REPEAT
       WRITELN(QOUT,'ENTER INDEX 1 - ',N_VARS:3);
       RST;READLN(QIN,N_V_T); G_V:=FALSE;
       IF(N_V_T IN (.1..N_VARS.)) THEN G_V:=TRUE;
       IF G_V THEN BEGIN
         WRITELN(QOUT,' INDEX=',N_V_T:3,' CORRECT?');
         G_V:=YES; END; (*IF G_V*)
      UNTIL G_V;
```

```
      END; (*ELSE N_VARS>1*)
   IN_V_T:=N_ELMS(.N_V_T.);
   IF((N_TEST=8) AND (IN_V_T<15)) THEN
     BEGIN
       WRITELN(QOUT,'BECAUSE THE NUMBER OF DATA ELEMENTS IN THE ');
       WRITELN(QOUT,'CHOSEN VARIABLE,',IN_V_T:3,', IS < 15, THE');
       WRITELN(QOUT,'THE PROGRAM CANNOT PERFORM THE CHISQUARED');
       WRITELN(QOUT,'GOODNESS OF FIT TEST.  CHOOSE AMONG:');
       REPEAT
         WRITELN(QOUT,'20  KOLMOGOROV: USER SPECIFIED PARAMETERS,');
         WRITELN(QOUT,'    NORMAL,EXPONENTIAL,UNIFORM,ERLANG,WEIBULL');
         WRITELN(QOUT,'21  LILLIEFORS:  PROGRAM COMPLTED PARAMETERS,');
         WRITELN(QOUT,'    NORMAL,EXPONENTIAL');
         WRITELN(QOUT,'22  SHAPIRO-WILKES:  PROGRAM  COMPUTED');
         WRITELN(QOUT,'    PARAMETERS, NORMAL ONLY');
         RST; WRITELN(QOUT,'ENTER CHOICE, 20-22');
         READLN(QIN,N_TEST);
         IF (N_TEST IN (.20..22.)) THEN G_V:=TRUE;
         IF G_V THEN   BEGIN
           WRITELN(QOUT,N_TEST:5,' CORRECT?');
           G_V:=YES; END;  (* IF G_V *)
      UNTIL G_V;
   END; (*IF N<15 AND N_TEST=8 *)
   IF ((N_TEST IN (.20,21,22.)) AND (IN_V_T >30))  THEN
    BEGIN
      WRITELN(QOUT,'BECAUSE THE NUMBER OF DATA ELEMENTS',IN_V_T:4);
      WRITELN(QOUT,'IS GREATER THE 30, THE PROGRAM WILL ');
      WRITELN(QOUT,'AUTOMATICALLY OFFER YOU THE CHISQUARE');
      WRITELN(QOUT,'GOODNESS OF FIT TEST');
      N_TEST:=8;
    END;   (* IN_V_T > 30 *)
    W_NTEST(N_TEST); WRITELN(OP,ONE:5);WRITELN(OP,N_V_T:5); NONE_EX;
END;  (* CHECK_N *)
(*----------------------------------------------------------------*)
PROCEDURE CHECK_FILE(VAR N_EQ_S,N_IN_F,N_OUT_F,N_VARS:INTEGER;
         VAR RIGHT_F,EQUAL_S:BOOLEAN;VAR N_ELMS:VECTORI);
   (*WRITES DATA NAME TO DETERMINE IF IT IS THE CORRECT FILE.
     IF SO, CHECKS IF VARIABLES OR TREATMENTS ARE OF EQUAL LENGTH*)
VAR  I,J,K:INTEGER;     F_NAME:STRING;
BEGIN
 OUTSET:=(.11..16.);
 I:=0;
 CASE (N_IN_F) OF
  129: BEGIN RESET(A,'NAME = LOCH1.A.A');
           REPEAT I:=I+1; READ(A,F_NAME(.I.));
           UNTIL (I=50) OR (F_NAME(.I.)='|') OR EOLN(A);
           READ (A,N_VARS); FOR I:=1 TO N_VARS
           DO READ(A,N_ELMS(.I.));END;
```

```
   130: BEGIN RESET(B,'NAME = LOCHI.B.A');
             REPEAT I:=I+1; READ(B,F_NAME(.I.));
             UNTIL (I=50) OR (F_NAME(.I.)='|') OR EOLN(B);
          READ (B,N_VARS); FOR I:=1 TO N_VARS DO READ(B,N_ELMS(.I.));END;
   131: BEGIN RESET(C,'NAME = LOCHI.C.A');
             REPEAT I:=I+1; READ(C,F_NAME(.I.));
             UNTIL (I=50) OR (F_NAME(.I.)='|') OR EOLN(C);
          READ (C,N_VARS); FOR I:=1 TO N_VARS DO READ(C,N_ELMS(.I.));END;
   132: BEGIN RESET(D,'NAME = LOCHI.D.A');
             REPEAT I:=I+1; READ(D,F_NAME(.I.));
             UNTIL (I=50) OR (F_NAME(.I.)='|') OR EOLN(D);
          READ (D,N_VARS); FOR I:=1 TO N_VARS DO READ(D,N_ELMS(.I.));END;
   133: BEGIN RESET(E,'NAME = LOCHI.E.A');
             REPEAT I:=I+1; READ(E,F_NAME(.I.));
             UNTIL (I=50) OR (F_NAME(.I.)='|') OR EOLN(E);
         READ (E,N_VARS); FOR I:=1 TO N_VARS DO READ(E,N_ELMS(.I.));END;
   134: BEGIN RESET(F,'NAME = LOCHI.F.A');
             REPEAT I:=I+1; READ(F,F_NAME(.I.));
             UNTIL (I=50) OR (F_NAME(.I.)='|') OR EOLN(F);
         READ (F,N_VARS); FOR I:=1 TO N_VARS DO READ(F,N_ELMS(.I.));END;
    END;(*CASE*)
     WRITELN(QOUT);FOR I:=1 TO 49 DO WRITE(QOUT,F_NAME(.I.));
     WRITELN(QOUT);
     WRITELN(QOUT,' IS THIS THE DATA SET YOU WANTED?');
     IF (YES) THEN RIGHT_F:=TRUE ELSE RIGHT_F:=FALSE;
     IF RIGHT_F THEN
       E_SIZE(N_VARS,N_ELMS,EQUAL_S,N_EQ_S);   (*J=FILETYPE OF DATA*)
     IF RIGHT_F THEN FILE_IO(NIF,N_IN_F);
  END;(*CHECK_FILE*)
 (*----------------------------------------------------------------*)
 PROCEDURE MENU(VAR STATE:TEST_N;N_VARS:INT;EQUAL_S:BOOLEAN);
 BEGIN
   STATE:=(.1..3.)+(.8.)+(.20..22.);I_P(.1.):=0;
   WRITELN(QOUT,' PICK AMONG THE FOLLOWING NON-PARAMETRIC TESTS: ');
   WRITELN(QOUT,'1. BINOMIAL');
   WRITELN(QOUT,'2. QUANTILE');
   WRITELN(QOUT,'3. COX-STUART TEST FOR TREND <IF 2 VARS,PAIR>');
   IF (N_VARS>1) THEN BEGIN STATE:=STATE+(.4,13,14,15,16,23,24.);
       WRITELN(QOUT,'4.MANN-WHITNEY');END;
   IF ((N_VARS>1) AND EQUAL_S) THEN
     BEGIN      STATE:=STATE+(.5..7.)+(.9..12.);
       WRITELN(QOUT,'5.SIGN <PAIR>');
       WRITELN(QOUT,'6.MCNEMAR <PAIR>');
       WRITELN(QOUT,'7.R*C CONTINGENCY TABLE <PAIR>');
       WRITELN(QOUT,'  (CHI-SQUARED TEST FOR INDEPENDENCE)');
     END;
   WRITELN(QOUT,'8.GOODNESS OF FIT, N>14 (CHI-SQUARE)');
   IF ((N_VARS>1) AND EQUAL_S) THEN BEGIN
```

```
      WRITELN(CCUT,'9.WILCOXON SIGNED RANK <PAIR>');
     WRITELN(CCUT,'10.RANK CORRELATION (KENDALL & SPEARMAN) <PAIR>');
      WRITELN(CCUT,'11.NON-PARAMETRIC LINEAR REGRESSION <PAIR>');
      WRITELN(COUT,'12.NON-PARAMETRIC MONOTONE REGRESSION <PAIR>');
     END;
  IF (N_VARS>1) THEN BEGIN
       WRITELN(COUT,'13.MEDIAN');
       WRITELN(COUT,'14.KRUSKAL/WALLIS');
       WRITELN(QOUT,'15.SQUARED RANK  TEST FOR EQUAL VARIANCE');
       WRITELN(COUT,'16.HARTLEY TEST FOR EQUAL VARIANCE');END;
  IF (N_VARS>2) AND EQUAL_S THEN BEGIN STATE:=(.1..24.);
       WRITELN(QOUT,'17.FREIDMAN OR DURBIN  <BLOCKED>');
       WRITELN(COUT,'18.QUADE     <BLOCKED>');
       WRITELN(COUT,'19.COCHRAN     <BLOCKED>');END;
  WRITELN(QOUT,'20 KOLMOGOROV: USER SPECIFIED PARAMETERS, N < 31');
  WRITELN(QOUT,'     NORMAL,EXPONENTIAL,UNIFORM,ERLANG,WEIBULL');
  WRITELN(QOUT,'21 LILLIEFORS: PROGRAM COMPUTED PARAMETERS,');
  WRITELN(QOUT,'     N < 31, NORMAL,EXPONENTIAL');
  WRITELN(QOUT,'22 SHAPIRO-WILKES: PROGRAM COMPUTED PARAMETERS');
  WRITELN(QOUT,'     N < 31, NORMAL ONLY');
  IF(N_VARS>1) THEN BEGIN
       WRITELN(QOUT,'23.SMIRNOV (2-SAMPLE)');
       WRITELN(QOUT,'24.CRAMER-VON MISES (2-SAMPLE)');END;
END; (*MENU*)
(*------------------------------------------------------------------*)
PROCEDURE WHICH_TEST(N_VARS:INT;EQUAL_S:BOOLEAN;N_EQ_S:INT;
   N_ELMS:VECTORI);
VAR N_PARAMS,N_TEST,I,J,K:INTEGER;
     STATE:TEST_N;
BEGIN
  REPEAT RST;
     MENU(STATE,N_VARS,EQUAL_S);
     WRITELN(QCUT,'ENTER # OF CHOSEN TEST, OR 0 TO SEE LIST AGAIN');
     READ(QIN,N_TEST);
  UNTIL N_TEST IN STATE;
IF (N_TEST IN (.8,20,21,22.)) THEN
   CHECK_N(N_VARS,N_TEST,IN_V_T,N_ELMS) ELSE W_NTEST(N_TEST);
  CASE N_TEST OF
       1:  BINOM(N_VARS,FALSE);
       2:  QUANTILE(N_VARS,FALSE);
       3:  COXSTU(N_VARS,N_EQ_S,EQUAL_S);
       4:  MANWHT(N_VARS);
       5:  SIGNTEST(N_VARS,N_EQ_S);
       6:  MCNEMAR(N_EQ_S,N_VARS,EQUAL_S,FALSE);
       7:  RC_CONT(N_VARS,N_EQ_S,FALSE);
       8:  GD_FIT(N_VARS);
       9:  WILCOXON(N_VARS,N_EQ_S);
      10:  CORREL8(N_VARS,N_EQ_S);
```

```
          11:  REGRET(N_VARS,N_EQ_S);
          12:  MONGREG(N_VARS,N_EQ_S);
          13:  MEDIAN(N_VARS);
          14:  KRUWAL(N_VARS);
          15:  SQUARANK(N_VARS);
          16:  HARTLEY(N_VARS);
          17:  FRIEDMAN(N_VARS,N_EQ_S);
          18:  QUADE(N_VARS,N_EQ_S);
          19:  COCHRAN(N_VARS,N_EQ_S,FALSE);
          20:  KOLMOG;
          21:  LILLIE;
          22:  WILKES(IN_V_T);
          23:  SMIRNOV(N_VARS);
          24:  CRAMVONM(N_VARS);
     END;
END;(*WHICH_TEST*)
(*----------------------------------------------------------------------*)
PROCEDURE NAMER(VAR F_NAME:STRING;
          VAR V_NAME:MATR; VAR N_VARS:INTEGER);
VAR  I,J,NV,UL,LL: INTEGER;
        BLANK:CHAR;
BEGIN
        FOR  I:=1  TO 49 DO  F_NAME(.I.):='  ';F_NAME(.50.):='|';
      WRITELN(QOUT,'ENTER NAME  OF DATA  SET');I:=1;RST;
      REPEAT
            I:=I+1;READ(QIN,F_NAME(.I.));
      UNTIL  (EOF(QIN) OR EOLN(QIN)  OR  (I=49));
           REPEAT
                WRITELN(QOUT,'  ENTER NUMBER--IN DIGITS,E.G.,3--');
                WRITELN(QOUT,'  OF  VARIABLES OR  TREATMENTS');
                RST;READLN(QIN,N_VARS);
                   IF  (N_VARS>15)  THEN WRITELN(QOUT,
                       'MAXIMUM  15  VARIABLES  OR  TREATMENTS');
           UNTIL  (N_VARS<=15);
           FOR I:=1 TO  (N_VARS+2)  DO
              FOR  J:=1  TO  50  DO V_NAME(.I,J.):='  ';
           FOR I:=1  TO  N_VARS  DO
              BEGIN
                J:=1;WRITELN(QOUT,'ENTER  NAME  OF  VARIABLE  NR  ',I);
                REPEAT
                    J:=J+1;READ(QIN,V_NAME(.I,J.));
                UNTIL  (EOF(QIN)  OR  EOLN(QIN)  OR  (J=50));
              END;
END; (* NAMER*)
(*-------------------------------------------------------------------------*)
PROCEDURE E_BY_VAR(VAR LIST:VECTOR;VAR N_ELMS:VECTORI;VAR N_VARS,
                        N_EQ_S:INT;VAR EQUAL_S:BOOLEAN);
VAR        NV,UL,LL,T_VARS,FILE_TYP,CH_ONE,J,I: INT;
```

```
BEGIN
(* THIS PROCECURE READS AND WRITES DATA BY VARIABLES *)
WRITELN(QOUT,' WHEN ENTERING DATA ELEMENTS <1, START WITH 0.');
     LIST_C:=0;VAR_C:=0;LIST_PTR(.0.):=1;
REPEAT(*UNTIL NOT RE-ENTER ALL DATA*)
      FOR NV:=1 TO N_VARS DO
         BEGIN
            REPEAT (*UNTIL THE VARIABLE IS GOCD*)
                VAR_C:=0;LIST_C:=LIST_PTR(.(NV-1).)-1;GOOD_VAR:=TRUE;
                REPEAT(* UNTIL ALL OF VAR IS ENTEREC *)
                     RST;
                     WRITELN(QCUT,' ENTER VALUES FOR VARIABLE ',NV);
                     LIST_C:=LIST_C+1;
                     VAR_C:=VAR_C+1;
                     READ (QIN,LIST(.LIST_C.));
                     WHILE NOT EOLN(QIN) DC
                         BEGIN
                             LIST_C:=LIST_C+1;
                             VAR_C:=VAR_C+1;
                             READ (QIN,LIST(.LIST_C.));
                         END;
                    WRITELN(QOUT,' ADD MORE ELEMS TC SAME VARIABLE?');
                    WRITELN(QOUT,' OR CHECK THIS VAR? OR GO TO NEXT?');
                     REPEAT
                         WRITELN(QOUT,
                         ' ENTER A(ADD),C(CHECK)CR G(GO ON)');
                          RST;READLN(QIN,CH_1);CH_ONE:=ORD(CH_1);
                    UNTIL ((CH_ONE=129) OR (CH_ONE=131) OR (CH_ONE=135));
                UNTIL (CH_ONE<>129);
           IF (CH_CNE=131) THEN
                BEGIN
                   WRITELN(QOUT,' NUMBER OF ELEMS ENTERED ',LIST_C);
                   WRITELN(QOUT,' NUMBER OF ELEMS IN THIS VAR ',VAR_C);
                   WRITELN(COUT,' ');
                   FOR I:=LIST_PTR(.(NV-1).) TC LIST_C DO
                       WRITE(QOUT,LIST(.I.):10:3);WRITELN(QOUT,' ');
                   WRITELN(QOUT,'WANT TO RE-ENTER THIS VARIABLE?');
                   IF (YES) THEN GOOD_VAR:=FALSE;
                END;
              N_ELMS(.NV.):=VAR_C;
              LIST_PTR(.NV.):=LIST_C+1;
          UNTIL GOCD_VAR;
        END;
(*    SUMMARY CF DATA INPUT STEP    *)
      WRITELN(CCUT,'YOL HAVE ENTERED',N_VARS:3,' VARIABLES, WITH THE',
          ' FCLLOWING NUMBER OF ELEMENTS IN EACH');
       WRITELN(COUT,'   VAR NR          NR OF ELEMENTS');
       FOR I:=1 TO N_VARS DO
```

```
         WRITELN(QOUT,I,N_ELMS(.I.));
      WRITELN(QOUT,' DO YOU WISH TO RE-ENTER  THE ENTIRE DATA?');
   UNTIL NOT(YES); (* RE-ENTER ALL DATA*)
(* PREPARE TO SEND DATA TO FILE *)
N_ELMS(.0.):=N_VARS;
FOR I:=(1+N_VARS) TO 15 DO N_ELMS(.I.):=0;
E_SIZE(N_VARS,N_ELMS,EQUAL_S,N_EQ_S);
END;(* E_BY_VAR*)
(*-----------------------------------------------------*)
PROCEDURE E_PAIRS(VAR LIST:VECTOR;VAR N_ELMS:VECTORI;
                 VAR N_VARS,N_EQ_S:INT;VAR EQUAL_S:BOOLEAN);
VAR   N_PAIRS,I,J:INT;
BEGIN
      REPEAT RST;WRITELN(QOUT,' HOW MANY PAIRS WILL YOU ENTER?');
         READLN(QIN,N_PAIRS);WRITELN(QOUT,N_PAIRS:5,' CORRECT?')
      UNTIL YES;N_EQ_S:=N_PAIRS;
      FOR I:=1 TO N_PAIRS DO   BEGIN
         RST;WRITELN(QOUT,'ENTER VALUES FOR PAIR',I:5);
         READLN(QIN,LIST(.I.),LIST(.(I+N_PAIRS).));  END; (*FOR I:=*)
      WRITELN(QOUT,'DO YOU WANT TO CHECK FOR POSSIBLE ERRORS');
      IF YES THEN FOR I:=1 TO N_PAIRS DO
         WRITELN(QOUT,LIST(.I.):10:3,LIST(.(I+N_PAIRS).):10:3, I:5);
      WRITELN(QOUT,'DO YOU WANT TO MAKE A CORRECTION?');
      IF YES THEN  REPEAT
         WRITELN(QOUT,'ENTER INDEX OF PAIR TO BE CORRECTED');
         RST;READLN(QIN,J);RST;
         WRITELN(QOUT,'CURRENT PAIR',J:5,'VALUES:',LIST(.J.):10:3,
              LIST(.(J+N_PAIRS).):10:3);
         WRITELN(QOUT,'ENTER CORRECT VALUES');
         READLN(QIN,LIST(.J.),LIST(.(J+N_PAIRS).));
         WRITELN(QOUT,'DO YOU WANT TO MAKE ANOTHER CORRECTION?')
      UNTIL NOT (YES);
      FOR I:=2 TO 16 DO N_ELMS(.I.):=0;
      N_VARS:=2;N_ELMS(.0.):=2;
        N_ELMS(.1.):=N_PAIRS;N_ELMS(.2.):=N_PAIRS;
      EQUAL_S:=TRUE;
END; (* E_PAIRS*)
(*--------------------------------------------------------------------*)
PROCEDURE NEW_DATA(VAR N_EQ_S,N_VARS,FILE_TYP:INT;
     VAR EQUAL_S:BOOLEAN;VAR N_ELMS:VECTORI);
VAR   NV,UL,LL,T_VARS,CH_ONE,I,J:INTEGER;
      BLANK:CHAR;
BEGIN
    BLANK:=' '; LIST_C:=0;
    FOR I:=1 TO 300 DO LIST(.I.):=0.0;
    RST;
    NAMER(F_NAME,V_NAME,N_VARS);
      REPEAT
```

```
          WRITELN(QOUT,' ENTER FILETYPE (A-F) TO WHICH DATA GOES');
          RST; READLN(QIN,CH_1);FILE_TYP:=ORD(CH_1);
      UNTIL (FILE_TYP IN (.129..134.));K:=FILE_TYP-128;
 IF (N_VARS<>2) THEN E_BY_VAR(LIST,N_ELMS,N_VARS,N_EQ_S,EQUAL_S);
 IF (N_VARS=2) THEN BEGIN
  WRITELN(QOUT,'IF BOTH VARIABLES HAVE AN EQUAL NUMBER OF ');
  WRITELN(QOUT,'ELEMENTS, THERE ARE TWO WAYS TO ENTER DATA:');
  WRITELN(QOUT,'(V) ENTER ALL DATA OF VARIABLE 1, THEN ALL OF 2,');
  WRITELN(QOUT,' OR (P) ENTER PAIRED DATA VALUES TOGETHER');
  WRITELN(QOUT);
  WRITELN(QOUT,'IF THE VARIABLES ARE NOT OF EQUAL SIZE');
  WRITELN(QOUT,'YOU MUST CHOOSE THE FORMER (V)');
REPEAT RST;
  WRITELN(QOUT,'ENTER P FOR BY PAIR OR V FOR BY VARIABLE/TREATMENT');
   READLN(QIN,H);I:=ORD(H)   UNTIL (I IN (.151,165.));
   IF (I=151) THEN E_PAIRS(LIST,N_ELMS,N_VARS,N_EQ_S,EQUAL_S);
   IF (I=165) THEN E_BY_VAR(LIST,N_ELMS,N_VARS,N_EQ_S,EQUAL_S);
   N_VARS:=N_ELMS(.0.);
 END; (*IF N_VARS <> 2*)
   LIST_C:=0;
   IF (N_VARS=1) THEN LIST_C:=N_ELMS(.1.);
   IF (N_VARS>1) THEN
         FOR J:=1 TO (N_VARS) DO LIST_C:=LIST_C+N_ELMS(.J.);
   FOR J:=1 TO (N_ELMS(.0.)) DO WRITELN(QOUT,N_ELMS(.J.));
CASE (FILE_TYP) OF
  129: BEGIN REWRITE(A,'NAME = LOCHI.A.A,RECFM=F,LRECL=80');
        FOR I:=1 TO 50 DO WRITE(A,F_NAME(.I.));WRITELN(A);
        FOR I:=0 TO 15 DO WRITE(A,N_ELMS(.I.):5);WRITELN(A);
        FOR J:=1 TO LIST_C DO
          BEGIN
             WRITE(A,LIST(.J.):10:3);
             IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(A);
          END;
        FOR I:=1 TO (N_VARS+1) DO   BEGIN
             FOR J:=1 TO 50 DO WRITE(A,V_NAME(.I,J.));
             WRITELN(A);END;
     END;
  130: BEGIN REWRITE(B,'NAME = LOCHI.B.A,RECFM=F,LRECL=80');
        FOR I:=1 TO 50 DO WRITE(B,F_NAME(.I.)); WRITELN(B);
        FOR I:=0 TO 15 DO WRITE(B,N_ELMS(.I.):5);WRITELN(B);
        FOR J:=1 TO LIST_C DO
          BEGIN
             WRITE(B,LIST(.J.):10:3);
             IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(B);
          END;
        FOR I:=1 TO (N_VARS+1) DO   BEGIN
           FOR J:=1 TO 50 DO WRITE(B,V_NAME(.I,J.));
           WRITELN(B);END;
```

```
      END;
131:  BEGIN  REWRITE(C,'NAME = LOCHI.C.A,RECFM=F,LRECL=80');
          FOR  I:=1  TO 50 DO WRITE(C,F_NAME(.I.)); WRITELN(C);
          FOR  I:=0  TO 15 DO WRITE(C,N_ELMS(.I.):5);WRITELN(C);
          FOR  J:=1  TO LIST_C DO
            BEGIN
               WRITE(C,LIST(.J.):10:3);
               IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(C);
            END;
          FOR  I:=1  TO (N_VARS+1) DO   BEGIN
             FOR J:=1 TO 50 DO WRITE(C,V_NAME(.I,J.));
             WRITELN(C);END;
      END;
132:  BEGIN  REWRITE(D,'NAME = LOCHI.D.A,RECFM=F,LRECL=80');
          FOR  I:=1  TO 50 DO WRITE(D,F_NAME(.I.)); WRITELN(D);
          FOR  I:=0  TO 15 DO WRITE(D,N_ELMS(.I.):5);WRITELN(D);
          FOR  J:=1  TO LIST_C DO
            BEGIN
               WRITE(D,LIST(.J.):10:3);
               IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(D);
            END;
          FOR  I:=1  TO (N_VARS+1) DO    BEGIN
             FOR J:=1 TO 50 DO WRITE(D,V_NAME(.I,J.));
             WRITELN(D);END;
      END;
133:  BEGIN  REWRITE(E,'NAME = LOCHI.E.A,RECFM=F,LRECL=80');
          FOR  I:=1  TO 50 DO WRITE(E,F_NAME(.I.)); WRITELN(E);
          FOR  I:=0  TO 15 DO WRITE(E,N_ELMS(.I.):5);WRITELN(E);
          FOR  J:=1  TO LIST_C DO
            BEGIN
               WRITE(E,LIST(.J.):10:3);
               IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(E);
            END;
          FOR  I:=1  TO (N_VARS+1) DO   BEGIN
             FOR J:=1 TO 50 DO WRITE(E,V_NAME(.I,J.));
             WRITELN(E); END;
      END;
134:  BEGIN  REWRITE(F,'NAME = LOCHI.F.A,RECFM=F,LRECL=80');
          FOR  I:=1  TO 50 DO WRITE(F,F_NAME(.I.));WRITELN(F);
          FOR  I:=0  TO 15 DO WRITE(F,N_ELMS(.I.):5);WRITELN(F);
          FOR  J:=1  TO LIST_C DO
            BEGIN
               WRITE(F,LIST(.J.):10:3);
               IF((C=(J MOD 8)) OR (J=LIST_C)) THEN WRITELN(F);
            END;
          FOR  I:=1  TO (N_VARS+1) DO   BEGIN
             FOR J:=1 TO 50 DO WRITE(F,V_NAME(.I,J.));
             WRITELN(F); END;
```

```
        END;
     END;
      WRITELN(QCLT,' NOW, DO A TEST WITH THIS DATA SET?');
      IF YES THEN   BEGIN
          FILE_IC(NIF,N_IN_F);
          WHICH_TEST(N_VARS,EQUAL_S,N_EQ_S,N_ELMS);
      END;
  END;(*NEW_DATA*)
  (*---------------------------------------------------------------------*)
  PROCEDURE CEL_TEST;
  VAR I,J,K,N_TEST,N_OUT_F:INTEGER;
  BEGIN
      N_IN_F:=138;
      FILE_IO(NIF,N_IN_F);
      NAMER(F_NAME,V_NAME,N_VARS);
        STATE:=(.1,6,7,19.);
        WRITELN(QOUT,' CHOOSE AMONG THE FOLLOWING TESTS:');
        WRITELN(COUT,'1. BINOMIAL, 6.MCNEMAR, 7.R*C CONTINGENCY');
        WRITELN(QOUT,'19. CCCHRAN');
        REPEAT RST; WRITELN(QOUT,' ENTER TEST NUMBER');
            READLN(QIN,N_TEST)   UNTIL (N_TEST IN STATE);
       W_NTEST(N_TEST);
       CASE N_TEST CF
           1: BINOM(N_VARS,TRUE);
           6: MCNEMAR;
           7: RC_CONT(N_VARS,99,TRUE);
          19: CCCHRAN(N_VARS,0,TRUE);
       END;
        FOR I:=1 TO 49 DO WRITE(OP,F_NAME(.I.));
        WRITELN(OP);
        FOR J:=1 TO N_VARS CO   BEGIN
            FOR I:=1 TO 49 DO WRITE(OP,V_NAME(.J,I.));
            WRITELN(CP); END;
   END; (*CEL_TEST*)
  (*-----------------------------------------------------------------*)
   PROCEDURE CH_NEWCA;
   BEGIN
          WRITELN(COUT,'YOU HAVE A CHOICE OF TWO METHCDS TO ENTER DATA:');
   WRITELN(COUT);WRITELN(QOUT,'     1. ENTER EACH VALUE OF DATA SET.');
   WRITELN(COUT,'       VALUES MUST BE NUMERICAL, ANC MEASUREMENT AT ');
   WRITELN(QOUT,'LEAST ORDINAL. ALL TESTS MAY BE PERFORMED THIS WAY.');
   WRITELN(COUT);
   WRITELN(QOUT,'     2. ENTER CNLY THE COUNT OF SUCCESSES/FAILURES');
   WRITELN(QOUT,'GR ROW/COLUMN CELL COUNT.');
   WRITELN(QOUT,'       DATA VALUES AND LEVEL OF MEASUREMENT MAY BE');
    WRITELN(QOUT,'NCMINAL, BUT COUNTS MUST BE INTEGERS >= 0.');
    WRITELN(QOUT,'THE FOLLOWING TESTS MAY BE PERFORMED THIS WAY:');
    WRITELN(QOLT,'BINOMIAL,MCNEMAR,R*C CONTINGENCY (CHI-SQUARE)');
```

```
   WRITELN(QOUT);
  END; (*CH_NEWDA*)
(*------------------------------------------------------------*)
PROCEDURE OLD_DATA;
BEGIN
   REPEAT  (*UNTIL RIGHT FILE*)
     REPEAT (*UNTIL CHOICE IS A TO F*)
        WRITELN(QOUT,
       ' WHICH DATA SET IS TO BE USED? ''A''-''F''');
        RST;READLN(QIN,CH_1);N_IN_F:=ORD(CH_1);
     UNTIL (N_IN_F IN (.128..135.));
   CHECK_FILE(N_EQ_S,N_IN_F,N_OUT_F,N_VARS,RIGHT_F,EQUAL_S,N_ELMS);
   UNTIL RIGHT_F;
   WHICH_TEST(N_VARS,EQUAL_S,N_EQ_S,N_ELMS);
END;  (*OLD_DATA*)
(*------------------------------------------------------------*)
BEGIN
     OUTSET:=(.11..16.);
     CELL_T:=(.1,2,6,7,19.); (*INDICES OF TESTS WITH CELLS*)
   RST; TERMOUT(QOUT);
   WRITELN(QOUT,'DO YOU WANT TO DO A TEST ON A PREVIOUSLY ',
      'ENTERED DATA SET?');
   IF YES THEN OLD_DATA
     ELSE    BEGIN
       CH_NEWDA;
         REPEAT RST;
            WRITELN(QOUT,'ENTER V FOR VALUE OR C FOR COUNT');
            READLN(QIN,CH_1); V_OR_C:=ORD(CH_1);
         UNTIL (V_OR_C IN (.131,165.));
       CELL_CT:=(V_OR_C=131);
       IF NOT(CELL_CT) THEN
            NEW_DATA(N_EQ_S,N_VARS,N_IN_F,EQUAL_S,N_ELMS)
         ELSE CEL_TEST;
     END; (*IF NOT OLD DATA*)      WRITELN(QOUT);
   WRITELN(QOUT,' TO PERFORM TEST, TYPE "CRUNCH" IN CMS ');
   WRITELN(QOUT);
END.
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22134	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	LTC Stephen J. Paek, Code 55Ph Department of Operations Research Naval Postgraduate School Monterey, California 93943	3
4.	Deputy Under Secretary of the Army (Operations Research) Room 2E621, The Pentagon Washington, D.C. 20310	1
5.	Deputy Chief of Staff for Operations and Plans Attn: DAMO-ZD (Mr. E.B. Sandiver III) Room 3A538, The Pentagon Washington, D.C. 20310	1
6.	Headquarters U.S. Army Training and Doctrine Command Attn: ATTG Ft. Monroe, Virginia 23651	1
7.	Combined Arms Operations Research Activity Attn: ATZL-CAC-A (COL A. West) Ft. Leavenworth, Kansas 66027	1
8.	Director U.S. Army Concepts Analysis Agency 8120 Woodmont Avenue Bethesda, Maryland 20814	1
9.	Director U.S. Army TRADOC Systems Analysis Activity Attn: Dr. W. Payne White Sands Missile Range, New Mexico 88002	1

10. CDR Charles Hutchins, Code 55 1
Department of Operations Research
Naval Postgraduate School
Monterey, California 93943

11. LTCOL Joseph Mullane, Code 54Mn 1
Department of Administrative Sciences
Naval Postgraduate School
Monterey, California 93943

12. LTCOL Michael Hester, Code MPI-20 1
Headquarters Marine Corps
Washington, D.C. 20310

13. MAJ Philip J. O'Brien 4
541 - 44th Street
Des Moines, Iowa 50312

14. Commanding General 2
(Attn: Dr. Drysbick)
Development Center, MCDEC
Quantico, Virginia 22342

207214

Thesis
O224 O'Brien
c.1 Nonparametric statistics test software package.

30347
22 APR 86

207214

Thesis
O224 O'Brien
c.1 Nonparametric statistics test software package.

AMERICAN BINDERY
TOPEKA KANSAS
2

thesO224
Nonparametric statistics test software p
3 2768 000 99913 0
DUDLEY KNOX LIBRARY